VIC rapport

ÅRGÅNG 3 NR 5/6

PRIS 20:- inkl moms

SKOLEXTRA
Comalskolan
TIDNINGSEXTRA
Rapport om utländska tidningar och böcker
MÄSSEXTRA
Rapport från Hannover- och Nordiska Mikrodatormässan



473–5



Kommunicera med VIC 64S

Det moderna sättet att få information

Kopplar du ihop ett universalmodem och ett teledataprogram med din VIC 64S så kan du kommunicera med videotexdatabaser via din telefonledning (75/1200 baud). Väljer du ett terminalprogram, kan du med samma modem "tala" med andra VIC 64 ägare eller koppla upp dig mot databassystem som kommunicerar med 300/300 baud (1 baud ≈ 1/10 tecken per sekund).

Med universalmodement kan du alltså välja vilken typ av kommunikation du vill ha. Med videotexprogrammet (Teledata 64) inkopplat kan du t ex koppla upp Datavision, Videodata, Prestel, Telebild, Postel, Micronet osv, osv.

I dessa baser kan du finna massor av information. Köpa en skjorta, sälja en bil, beställa campingplats, skicka meddelanden m m, m m...

Några databaser:

Viewdata AB	08-743 06 65
Datavision	018-901 00
Telebild	08-13 58 00
Videotex telematris	08-99 44 30
Wettergrens	031-11 21 34

Kommunikation är en del av vad VIC 64S kan. I foldern "VIC till vardags" får du mer information om vad VIC kan göra för dig i vardagslivet.

Jag vill ha foldern "VIC till vardags"

Namn ____________________

Adress ____________________

Postnr/Ort ____________________

Sänd in kupongen till handic electronic, Box 1063, 436 00 Askim.

handic electronic ab

Box 1063, 436 00 Askim/Göteborg, Tel. 031/28 97 90
– ett företag i Datatronicgruppen –

Ledaren

Nu är sommaren äntligen här med ett långt skönt sommarlov. Eller tillhör du den gruppen som aldrig tar chansen att njuta i en hängmatta. Kanske har du sommarjobb eller fast jobb. VIC rapport kommer i alla fall att ta ledigt i juli och nästa nummer kommer först i augusti. Men för att kompensera er för detta så presenterar vi här ett riktigt tjockt dubbelnummer med massor av läsning.

Detta numret av VIC rapport är speciellt av flera anledningar. För det första vill redaktionen hälsa Åke Fredriksson välkommen tillbaka. Han kommer att ansvara för vår nya Comalskola samt för allt som har med programspråket Comal att göra.

Förutom Comalskolan fortsätter vi naturligtvis med de tre andra skolorna d v s Assemblerskolan, Nybörjarskolan och Forthskolan. För det andra har vi två reportage från de stora datamässorna som har ägt rum. En i Tyskland, Hannovermässan och en i Sverige, Nordiska Mikrodatormässan. På dessa mässor fanns alla de nyheter som vi kan vänta oss på marknaden i den närmsta framtiden. Utförliga reportage hittar du längre fram.

Trevlig läsning och trevlig sommar!!

Innehållsförteckning

VIC redaktion: Box 52054, 12612 Stockholm, Telefon 08-744 59 20

Ansvarig utgivare: Nina Linander

Administrativ redaktör: Elisabeth Höglund

Teknisk redaktör: Matts Nilsson

Layout: Lennart Hammarstad

Övr. medarbetare: Åke Hedman, Joakim Aspengren, Ola Johansson, Åke Fredriksson

Annonser: Lasse Brolin, Hagalund, 27035 Blentarp, Telefon 0411-473 42

Tryck: *MINAB/GOTAB*

ISSN-NR 0281-8043

Annonsorder och annonsmaterial (helorginal eller negativ film) enligt överenskommelse.
Tryckförfarande. Offset
Upplaga. 22000 ex.

Nr 7 av VIC-rapport utkommer preliminärt den 13 augusti. Manusstopp för nr 7 är den 9 juli och annonsstopp den 16 juli.

Manusstopp för nummer 8 är den 8 augusti och annonsstopp den 13 augusti.

Hannover Ce-Bit mässan

— giganten bland mässor

av Bengt Litnäs

Tänk dej Älvsjömässan! Stoppa in den två gånger i hall 1 i Hannover och fyll på med datorer.
Tänk dej Råsunda fotbollsplan! Lägg till 99 planer och du får det totala utställningsområdet.
Lägg till ytterligare 110 planer och du har parkeringsområdet för 60000 bilar.

Till all lycka var endast 5 hallar fyllda med datorer och annan elektronik till nytta eller nöje. Alla de andra hallarna var fyllda med varor från synålar till lastbilar. Två hallar var t ex fyllda med belysningar. Åtskilliga tusen lampor var tända och hetta motsvarande fullt vad som fordrades för en god bastu.

Mässans stora dragplåster var onekligen elektroniken. I fem hallar med tillsammans 128000 m² yta ställde ca 1200 utställare ut sina produkter. 2/3 därav var tyska utställare. Detta visar att Västtyskland är på snabb frammarsch vad gäller elektronik. Bara för några år sedan var det ett U-land. År 1983 besöktes denna mässa av ca 300000 besökare. I år noterades efter tre dagar 130000.

Do-it-yourself

En utställning i utställningen var "microtronic", där man på 10800 m² visade allt vad som fanns i elektroniska pryttlar för självbyggare. I det nordtyska TV-programmet NDR III går sedan i januari en serie, i vilken man bygger en ganska avancerad dator med en Z80-processor, Thomsons nya grafikchip samt med tilläggskort för Motorolas 68008. En ganska dålig marknadsföring har gjort att idén inte slagit särskilt väl ut.

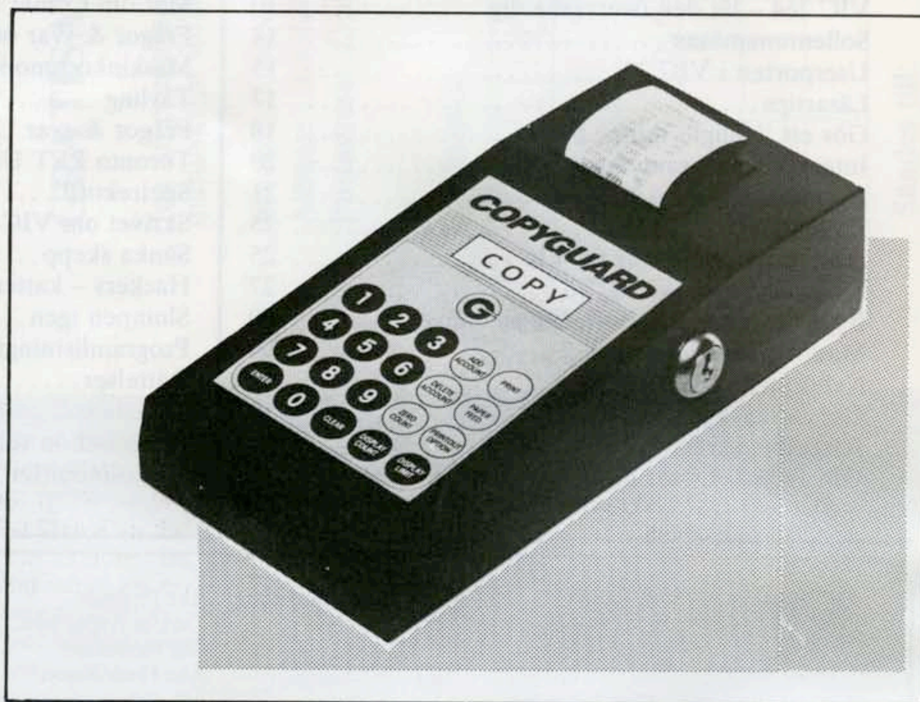
Utbudet av självbyggen är stort från datortidskrifternas sida. Här kan särskilt nämnas c't — en högkvalitativ tidskrift och med en självbyggdator med ett 8086-chip samt 128 KB RAM utbyggbart till 256 KB. Denna dator kommer efterhand att byggas ut för att motsvara en högmodern apparat för högt ställda krav. Priserna för dessa datorer ligger som byggsats medd 4000 till 6000 kronor.

I början var ordet...

och ordet intog en dominerande ställning på mässan. Alltmer börjar företagen förstå, att ord egentligen är små, kostsamma ting, vilka man helst ska behandla automatiskt.

Till detta område hör ju egentligen allt. Datorer, skrivare, disketter, bildskärmar, diktafoner, telefoner med modem, högtalare, läsare (OCR), software. Här återfanns också de riktigt stora utställarna — de som erbjöd kompletta paketlösningar. Här fanns Grundig, Philips, Sony, Sanyo, Panasonic, Sharp, LM Ericsson, Triumpf-Adler m fl.

Texas visade t ex att man kunde tala med sin dator via mikrofon. Man hade gjort ett enkelt program, där bildskärmen visade 10 färger och genom att säga färgen flyttade sig cursorn till nästa färg.



En vakthund för kopieringsapparater — kanske något för våra storföretag.

Men man måste vara noggrann med uttalet. Violett uttalas ju med "v" i Sverige, men i Tyskland som "f". Jag gjorde just det felet och datorn förstod inte vad jag menade. Detta är på samma gång en svaghet och en styrka. Om ca fem år är förmodligen tekniken så långt kommen, att man kan tala med sin dator i stället för att hacka. Nu är det emellertid så, att en människan röst i stort sett är lika säregen som ett fingeravtryck.

Genom att renodla ett röstspektrum kan man göra delar av ett stort datasystem röstspecifikt, d v s man kan i ett företag skilja åtkomligheten av känsliga data till endast vissa betrodda personer.

Skrivare fanns i Babylonien...

...men de skiljer sig en aning från dagens skrivare till en dator. Ett enormt utbud



står till förfogande — det är bara att välja! Eftersom VIC rapport berättar om VIC-20 och VIC-64 ska vi kanske begränsa oss till skrivare som skulle passa för dessa — inte minst med tanke på priset.

Marubeni Corp. i Tokyo visade en skrivare med goda egenskaper. Bokstavsmatriserna har en punkttäthet av 9×9 och hastigheten är 120 tecken/s vid textskrivning samt 5 600 punkter/s vid grafik. Två skrivriktningar vid text samt en väg vid grafik. Svenska särtecken finns inbyggda. Maskinen skriver med olika täthet och olika storlekar på bokstäverna. Maskinen heter COPAL SC-1000 och priset i Västtyskland är DM 1 000 = 3 000 kronor. Parallellt och seriellt gränssnitt ingår som standard.

Mannesmann-Tally gör en skrivare som heter MT-80 och har en 9×8 p matris, gör 80 t/s, grafikmöjlighet, tvåvägsskrivning. Denna maskin har fått utmärkta lovord i tysk fackpress och kostar med V-24 gränssnitt ca 3 300 kronor.

Seikoshas skrivare GP 100 VC är prisvänlig och kostar i Västtyskland ca 2 100 kronor. Det är en vettig maskin utan alltför många finesser.

En stjärna bland skrivare är STARS delta-10, vilken är en ren önskedröm. Med en 9×9 matris som standard och 8 KB buffertminne, med 160 tecken/s, med 6 olika bokstavsstorlekar, med olika grafiktäthet — 480/960/1920 punkter — är det en maskin runt 1 500 DM.

Mitsubishi visade sin färgskrivare M4234, vilken åstadkommer 4-färger med ett värmeskrivhuvud. Vid 2-vägsskrivning kan ytterligare tre färger tryckas. Hastigheten är 180 p/s och vid grafik skriver den 24×1440 p/rad. Skriften är angenäm att läsa.

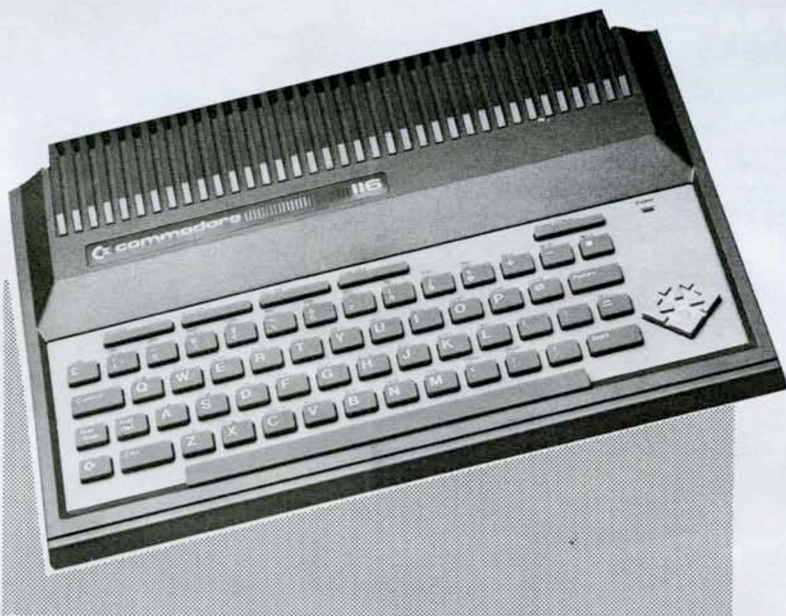
Vi kanske också skall se litet på en östtysk skrivare vid namn EUROPRINT K6311 FT/P för DM 999,—, FT/S för DM 1 099,— samt en helt ny K6312 t-P för DM 1 500,—. Den har testats av västtyska tidningar som i stort ansett den bra, men med sedvanliga erinringar om vissa tekniska besvärligheter vid byte av färgband etc. I grundversion är de alla utrustade med parallellt gränssnitt och mot kraftiga tillägg får man ett V-24. Skrivarna har en 9×7 matris med en täthet av 100 punkter/tum. Normal-, tät-, stor- och kursivskrift kan åstadkommas och skriften är väl läsbar.

För att återkomma till gränssnitten, så missgynnas alltid Commodore-datorer. Trots att VIC-20 ligger som nr 2 och VIC-64 som nr 1 på 10-i-topp-listan, så görs praktiskt taget inga periferier med V-24 gränssnitt. Det är alltså ett sätt att låta konsumenten (du) betala två gånger. Vi skall återkomma till denna fråga på annat ställe.

Skrivmaskin som skrivare?

Ja, varför inte? Den som sätter värde på en skönskrivare och kan undvara grafiken gör inget dåligt val. Dessutom slipper man den ofta tråkiga tangentuppsättningen på datorn. Skrivmaskinen köper man med å, Å, ä, Ä, ö, Ö och genom program definierar man dessa tecken i datorn.

Helt ny på mässan var Triumph-Adlers Gabriele 9009, vilken är ersättare för Gabrielle 8008. Måttliga 13 tecken/sek gör att en A4-sida tar ca 5 min att skriva. Men då är det lika snyggt som om det vore gjort manuellt — kanske snyggare. En massa finesser är inbyggda, såsom tabuleringar, understrykningar, jämn högereller vänstermarginal, lift-off, 4 radavstånd m m, m m. Det bästa kommer sist: i september kommer att lågpris gränssnitt →



Commodore 116 — ny design och ny cursorstyrning. Kommer troligen inte att lanseras i Sverige.

för V-24 och då är det bara att koppla ihop din Commodore med Gabriele. Priset på skrivaren är 998,— DM. För Olivetti- och Olympiamaskiner finns det hårdvara, vilken man själv kan bygga in för att göra dessa kompatibla med Commodore.

Diskettenheter

På detta område sker en enorm utveckling. Inte nog med att man bygger disketten slim-line, dvs ca 60 mm hög, de blir också allt mindre och mindre, men samtidigt lagrar de mer och mer data. Sony slog till förra året med sin 3 1/2-tums diskett och nu ligger en ny modell med en lagringskapacitet av 1 MB framme för produktion.

Som vanligt med ny teknik, så kommer en massa olika format på en gång. Seikosha gör en diskett med 3 1/4-diskar. Även 3-tum och 4-tum finns. Nu har man emellertid bestämt sig för att Sonys 3 1/2-tum ska bli standard.

Hemdatorer

Ett stort utbud visades. men de var inte alltid så lätta att hitta bland alla kontorsdatorer. Till all lycka kom jag in genom ingång nordost till hall 1 och ramlade direkt på en tjeckoslovakisk dator. Den heter SMEP—01 och processorn heter MHB 8080 A 2 MHz med 65 KB. Den har nu inget att göra med Intels 80-serie utan är en variant av Z80. I ROM finns 16 KB. Datorn har ASCII-tangentbord och är även avsedd för grafik med 256×256 punkters upplösnings. När man försöker läsa reklambroschyren på engelska, så är det nästan omöjligt att fatta vad det är tillverkaren vill tala om för läsaren. Samtal med utställningspersonalen är på grund av språksvårigheterna ej heller meningsfulla. Alltnog, denna dator för oss ca 10 år tillbaka i tiden, dvs nästan stenåldern.

I DDR har man också gjort en hemdator, vilken presenterades på Leipzigmässan nyligen. Nu fanns den också i Hannover och den liknar mer det man är van vid. ROBOTRON 1715 heter den. Den har en U 880 processor 2,5 MHz med 64 KB som standard. För bildskärmen står såtliga 64×16 tecken/rad till förfogande, vilken kan utvidgas till 80×24.

Gränssnitten är 1× seriell för skrivare samt 1× V-24. Konversationsövningar kan föras på följande språk: BROS, JAMB, PASCAL, BASIC, MABS 1520 samt även CP/M-drift.

Två diskettstationer står till förfogande på vardera 128 KB. Priset är DM 4500,— för dator, ett diskettverk samt monitor.

Den nya trenden för hemdatorer ligger i bärbarheten. Epson och Olivetti har gjort små apparater att bära i en portfölj. Hitachi kommer med en som heter MB-H1 på 64 KB. Den är av den nya MSX-typen. Vikt 3,4 kg och måtten 324×55×229 mm. SORT gör en IS-11 med 32 KB-RAM och 64 KB-ROM. Apparaten väger 2 kg och har måtten 300×48×215 mm. Den kan förses med en liten skrivare på vänster sida och på höger sida med en sifferenhet. En mikrokassett är inbyggd. LASER 3000 är en läcker nykomling. Centralprocessorn heter 6502 A och för styrning finns en 8048. 64 KB RAM expanderbart på kortet till 192 KB samt 24 KB Basic-ROM medföljer. Tillbehör: joysticks, diskett, kassettdäck, ljuspenna, färgplotter — allt i en läcker benvit färg.

Äntligen Commodore!

Det bästa har jag gömt till sist. De efterlängtade nyheterna — de tre nya musketörerna, som inte alls ersäter VIC-20 eller VIC-64, utan ger nya valmöjligheter.

Commodore C 16

Deta är VICs större broder. Den har 16 KB RAM, varav 12 KB står till förfogande för programmering. I ROM gömmer sig hela 32 KB för styrsystem och Basictolk. Grafiken är 25 rader med 40 tecken eller 320×200 punkter. Nytt är att bildskärmen kan delas upp och text och grafik användas samtidigt. Ljudet åstadkommes av en tongenerator och en brusgenerator med åtta ljudlägen.

Commodore 116

En helt ny design kännetecknar denna dator som innehållsmässigt är helt lik C 16 med ett undantag — cursorstyrningen är ny.

Commodore 264

Ser ut som 116 med har 64 KB RAM på minnet, av vilka 60,8 KB står till användarens förfogande. 32 KB-ROM för drift- och Basicsystemen. Detta ROM kan utvidgas med ytterligare 32 KB för viss software. Grafiken — utan sprites — är lika C 16 men ljudet har förbättrats genom att en ljudgenerator lagts till. Här finns också en maskinspråksmonitor inbyggd, liksom ett fritt valt användarprogram, t ex ordbehandling eller liknande.

Gemensamt för alla är den nya processorn 7501 med 0,89—1,76 MHz — ej kompatibel med 6510 i C-64. I den tyska broschyren står att tangentbordet har 67 tangenter samt åtta funktionstangenter.

Den nya Basicen har 75 kommandon, vilket är en avsevärd förbättring. Anslutningarna är USER PORT och seriellt gränssnitt (= C-64), en modulingång, två



Commodore 264 — utvidgad BASIC, extra ljudgenerator, inbyggd maskinspråksmonitor — detta är bara några av de finesser man kan finna på 264:an. Lanseras i Sverige tillsammans med J64:an i slutet av året.

joystickingångar samt en ingåt för Datasetten (ej 64). Dessutom ingångar för TV och monitor, audio in- och ut, samt strömförsörjning.

Till de nya datorerna kommer också nya tillbehör. Den nya Datasetten heter 1531 och den nya disketten, som är ca 6 ggr snabbare heter SFS 481. Denna skall modifieras för att även arbeta med C-64.

Commodore 364???

På mässan fanns inte modell 364. Men likväl skriver den tyska pressen om denna modell. För det första är den större, 420 × 65 × 240 mm mot 264:ans 336 × 65 × 197 mm. Därtill ska den har 48 KB ROM mot 32 samt ha en inbyggd talmodul med 250 ord som standard. Tangentbordet är större och har 86 tangenter mot 264:ans 67.

Till nyheterna hör också den nya skrivaren VC 1526. Denna har den nyheten att den även kan drivas som en VC 1525. Emellertid måste man då göra ingrepp i hårdvaran och att ledningen från PIN 16 till U4D i stället lägges om till jord. Commodore själva påstår att VC 1526 ej har grafikmöjligheter. Däremot har använda-re en annan mening därom och påstår att grafik är möjlig.

Ytterligare en skrivare MPS 801 är en något ändrad version. Sekundäradresserna 0 och 7 har bibehållit sin betydelse, de övriga ej.

Den nya färgmonitorn från Commodore var ganska läcker och hade en upplösning som får betraktas som hög. Priset i Västtyskland ligger idag på DM 765,—, vilket får betraktas som mycket billigt.

En nyhet bland "innefolket" är att C-64 skall bli utrustad för bildskärmstext. Detta sannolikt för att ta upp konkurrensen med Blaupunkt, Saba, Philips och Grundig, vilka alla erbjuder paketlösningar inkl dator till låga priser.

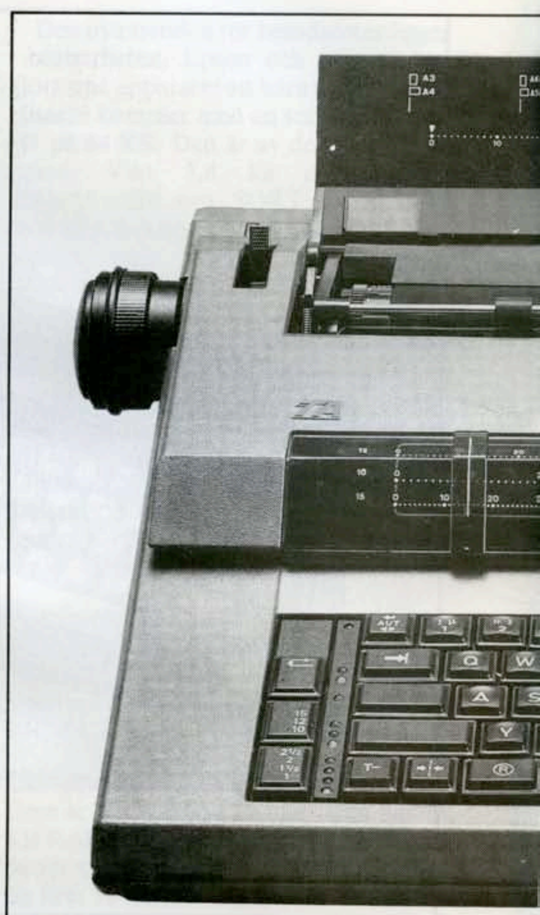
I Västtyskland ligger bland hemdatorerna som nr 1 C-64, som nr 2 VIC-20. Ca 375000 av dessa datorer lär vara sålda. Bland kontorsdatorerna ligger Commodore 80-serie 1:a.

För Commodores hemdatorer kommer det ny mjukvara. Sålunda ligger för de nya datorerna ett ordbehandlingsprogram klart, ett tabellkalkylerings-, ett grafik- samt ett förvaltningsprogram.

Vidare kommer ett program som heter Magic-Desk och är en motsvarighet till Koala teckningsbord. För 64:an skall komma en serie läroprogram med i huvudsak matte som ämne. Vidare kommer →



C-16 — storebror till VIC — i vilken bildskärmen kan delas upp och text och grafik kan användas samtidigt. Kommer troligen inte att lanseras i Sverige.



Gabriele 9009 från Triumph-Adler — skriver långsamt men mycket snyggt — ett alternativ för dig som kan undvara grafiken.

Autoprint — med denna enhet inkopplad i bilen kan man ständigt nås av skrivna meddelanden.



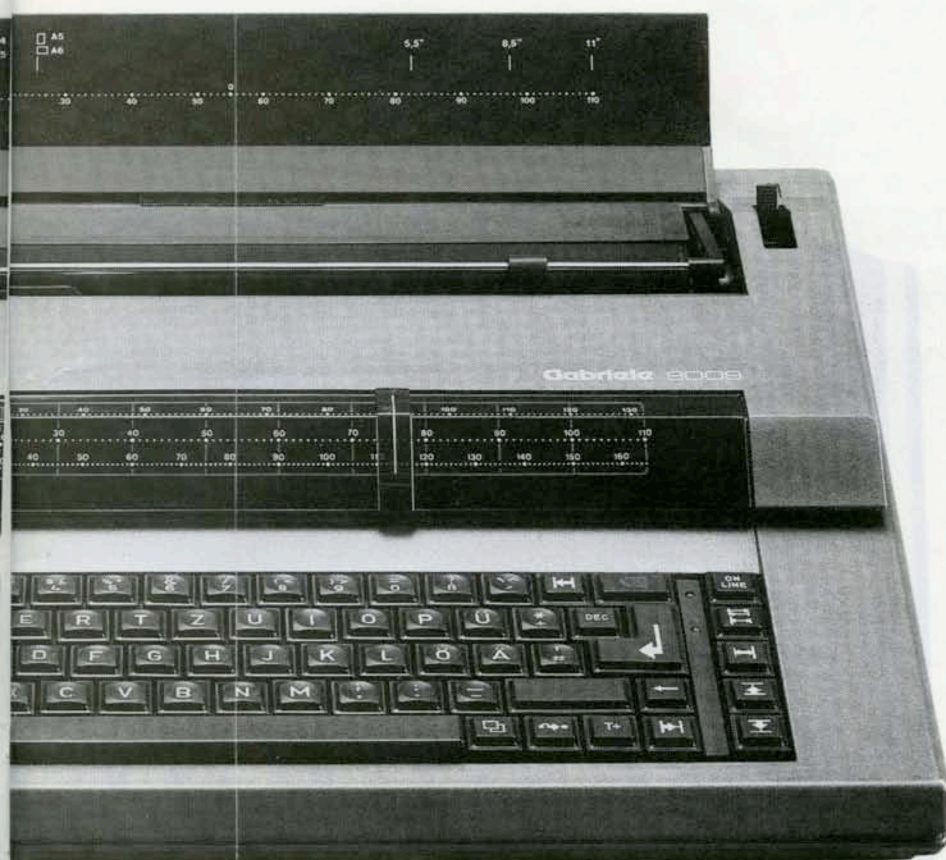
för barn mellan 4—6 år 40 program på en diskett för att lära sig läsa och skriva. Nya spel är Soccer International, Viduzzles, Jack Attack och Solar Fox samt en elektronisk kokbok.

Men det skall också komma hårdvara för att expandera t ex 64:an. Cardboard 5 är en utvidgning av modulingången med fem (5) ingångar. Ljuspenna, gränssnitt m m kommer också, liksom nya diskstationer från MSD. Från Inkwell Systems kommer ett grafikprogram Flexidraw med ljuspenna. Ett tilläggsprogram Penpal möjliggör överföring av bilder via modell till andra 64:or.

Kanske går det mot ljusare tider för Commodore-hackers.

Australien
är kanske inte det land man först tänker på när det gäller elektronik. Landet som koloniserades av från England deporterade förbrytare. Landet där idag anor från dessa förbrytara är lika med adelskap — nästan.

På mässan fanns ett par ovanliga tillämpningar av elektronik.



En vakthund för en kopieringsapparat t ex. Man bygger in en liter dosa i kopieraren och man kan kopiera endast om man "hackar" in ett godkänt nummer. Ganska listigt och vore säkerligen något för det svenska statsverket eller varför inte för Journalistförbundet?



Exempel på en 3 1/4-tums diskett från Seiko. Den nya standarden på slim-line disketter har dock satts till 3 1/2-tum.

En annan pryl är Autolink och Autoprint. Den senare har man i bilen på resor och man kan då nås av skrivna meddelanden upp till 255 tecken med 40 tecken/rad. Överföringen är diskret, d v s bara föraren kan ta emot det och han behöver inte ens vara i bilen när meddelandet kommer.

Autolinken har en komplett tangentuppsättning samt en LCD-tavla på vilken 4 × 40 tecken samtidigt kan visas. Totalt kan 1500 tecken överföras. Denna kommunicerar båda vägarna. Genom kodord och kodering kan läsning omöjliggöras för utomstående.

Det är väl sannolik nödvändigheten som tvingat fram dessa båda nyheter, genom de enorma avstånden i en delvis mycket ogästvänlig terräng.

Vet du var det värsta med Hannovermässan är? Du lär känna din egen ofullkomlighet! Det finns så mycket att se och lära, men din tid och din ork räcker ej till. Men ännu värra är, att nästa år så slår dom till igen med en ny och ännu större mässa. Träna maraton, så går det bättre. ●

Här intill presenterar vi en av Commodores nyheter, som presenterades på den stora datormässan i Hannover. Denna artikel har vi fått från vår reporter i Tyskland och all information är hämtad därifrån. Vi har ännu inte fått reda på vad datorn kommer att heta i Sverige. Vi har försökt ta reda på detta genom Handic Electronik men det verkar som om man inte har bestämt sig för något namn ännu. Vi har därför bestämt oss för att kalla den VIC 264. Det ska i sammanhanget påpekas att VIC rapport förbehåller sig alla uppgifter som står i artikeln. Vi vet inte om dessa kommer att överensstämma till 100 % med de uppgifter som kommer i samband med lanseringen i Sverige. Troligtvis kommer VIC 264 att släppas på den svenska marknaden i slutet av 1984.



VIC 264 — Låt den överraska dig!

av Bengt Litnäs

Efter allt skäll VIC 64 fått för sin magra BASIC har nu Commodore låtit kusinen VIC 264 få ett riktigt trevligt språkelement. Nu kan du äntligen utnyttja datorns inne-



Därtill kommer den nya möjligheten att få en mjukvara inbyggd redan vid köpet. Följande program kan väljas

1) **Magic desk** en simulering av ett skrivbord och ska kunna skriva brev, räkna, skiftväxlingsförvaring, tidgivning mm. Ett program för hemmet.
2) **SUPERSCRIPT** ett halvavancerat textsystem.
3) COMMODORE 3-PLUS-1 är en kombination av text och adressbearbetning, tabellkalkylation samt grafik med fönsterteknik.
4) **LOGO** ett lättlärt programeringsspråk.
5) **PILOT** ytterligare ett programmeringsspråk.
6) **EASYCALC 264** ett program för tabellkalkylering.
7) **COM 264**
8) **FINANCIAL ADVISOR** ett ekonomiprogram.

Då ett av dessa program ingår i köpet måste alltså köparen på förhand bestämma sig för hur han vill använda datorn. Över funktionstangenterna kan dessa program lätt kopplas in eller ur. Övriga program ovan kan köpas som moduler och, i därför särskilt avsedd ingång på baksidan, göras direkt tillgängliga.

Det bör obserberas att det inbyggda programmet ligger i ROM-minne och vill man byta, så måste datorn öppnas och då har man förverkat garantin.

264:ans BASIC

Som redan sagts så har de nya datorerna fått en förbättrad BASIC. Grafik-, ljud- och diskettkommandon med hjälpfunktioner kan nu styras över den inbyggda BASICen.

Alla kommandon, vilka kan direktadresseras kan även utföras över funktionstangenterna och över dessa programme- →

boende, stora egenskaper helt utan att behöva ta din tillflykt till toolkits, assembler, Simon's etc. Med VIC 264 har Commodore flyttat fram sina positioner utan att därför på något sätt överflygla andra konkurrenter med en redan bra BASIC. Men lägger vi till de inbyggda maskinspråket, assembler och disassembler, så rycker VIC 264 fram som en mycket kraftfull dator, vilken även är lätt att umgås med.

ras. De fungerar på samma sätt som KEY-kommandot hos Simon's BASIC.

Med DISPLAY kan kommandona påkallas. En särskild egenhet har HELP funktionen. Om ditt program avbrytes av ett felmeddelande, så trycker du bara på HELP-tangenten (t ex f8) och då visas den felaktiga raden reverserat. Däremot meddelas ej exakt var felet finns på denna rad som t.ex. fallet är med EXBASIC LEVEL II. Denna hjälp får man normalt även med hjälp av BASIC:en, varför omvägen över HELP är mindre lätt att förstå.

Grafik
Bildskärmen kan få tre olika modem:
1) den högupplösande grafiken med 200 × 320 punkter
2) flerfärgsmodem med 160 × 200 punkter samt
3) textmodem med 40 × 25 rader.

Alla tre framställninssätten kan utföras samtidigt och det måste betraktas som ett extra plus. Styrningen sker över GRAPHIC kommandon och löses på följande sätt:
1) Modem 0 = textframställning och är standard vid inkopplingen av 264: an.
2) Modem 1 = högupplösande grafik
3) Modem 2 = högupplösande grafik plus text i de understa fem raderna.
4) Modem 3 = flerfärgsgrafik samt
5) Modem 4 = flerfärgsgrafik med text i den understa fem raderna.

I ställning 4 och 5 kan alltså en högupplösande (HIRES) bildskärm kombineras med fem rader text. Därigenom kan man överst på skärmen visa grafik och därunder t ex genom LIST låta en listning passera eller en tabell.

Genom att välja ett av grafikmodem 1—4 reserveras en 10 KB minnesarea i RAM.

VIC 264 kan alltså framställa 128 färger eller rättare 16 färger och varje färg i åtta toner eller nyanser. Dessa färger kan direktadresseras antingen med CONTROL-tangenten eller genom enkla BASIC kommandon i programmet.

Färg- och grafikmodem
Region, färg, färgstyrka, Region (0 till 4)
0 = färgstyrka
1 = teckenfärg
2 = flerfärg 1
3 = flerfärg 2
4 = ramfärg
Färg (1 till 16)
Färgstyrka (0 = mörk till 7 = ljus)

Grafik
modus, radera modus (0 till 4)
0 = text
1 = HIRES
2 = HIRES + text (delad bildskärm)
3 = flerfärg 1
4 = flerfärg 2 (delad bildskärm)
radera (0 eller 1)
0 = ej radering
1 = radera

VIC 264 har delvis ganska kraftiga grafikkommandon. Det är inte längre tal om spritar — de existerar ej hos VIC 264. Man kan dock välja ut bestämda områden hos bildskärmen eller förflytta dem, utan att behöva inskränka sig till VIC 64:ans begränsade storlek av 24 × 21 punkter.

Däremot är oklart, om man genom kollisionsfrågor kan åstadkomma något liknande spritar för för- och bakgrundsfärger samt alla de möjligheter man genom dessa når för att konstruera spel och liknande. Ur BASICordboken eller handboken kan detta inte utläsas.

För att återkomma till handboken, så har den vanligtvis varit föremål för kritik, men för 264:an har det blivit en riktigt bra handbok. Alla BASICkommandon är utförligt förklarade till och med finns talrika exempel på användningen.

Grafikkommandon

CHAR	= fogar in text direkt i HIRES-grafik
BOX	= tecknar rektanglar
CIRCLE	= tecknar cirklar, ellipser
PAINT	= fyller en bestämd area med en bestämd färg
SCALE	= Bits-mönstrets skalförändring vid HIRES eller flerfärgsgrafik.
DRAW	= tecknar punkter och linjer
LOCATE	= fastställer utgångspunkten för teckenkommandon
SSHAPE	= lagrar en förutbestämd del av HIRES grafik på kasset eller diskett
GSHAPE	= Den från SSHAPE avsedda delen kan man med GSHAPE återföras till valfritt ställe på bildskärmen. GSHAPE tillåter flera modem: * = såsom data är lagrade 1 = revers framställning 2 = OR-länkad med omgivningen 3 = AND-länkad med omgivningen 4 = XOR-länkad med omgivningen
RCLR (N)	= Tilldelar en bestämd bildskärmsarea N (0—4) en bestämd färg (areor: se under COLOR)
RDOT(N)	= Anger grafikcursorns aktuella ställning.
RGR(N)	= Man bibehåller det grafikmodem i vilket man befinner sig
RLUN(N)	Tilldelar färgarean N anvisad färgstycke

Ljud
264:ans ljudmöjligheter är mycket begränsade. Endast två kommandon står till förfogande:

VOL (0 till 7)	= reglerar ljudstyrkan hos frambringad ton
SOUND x, y, z	= frambringar en ton alt. ett brusljud x (1 till 3) 1 = hög stämma från mycket högt till medellåg 2 = låg stämma från medelhög till mycket låg 3 = brusljud (t. ex. åska) y (0 till 123) = notvärden, (0 = låg ton, 1023 = hög ton) z (0 till 65535) = tonlängden

I förhållande till VIC 64 är således ljudmöjligheterna starkt begränsade. Nu är det emellertid ej heller frågan om att ersätta VIC 64 utan att bredda sortimentet.

Hjälp (vid programmering)
Dessa funktioner är däremot förstklassiga. För redigering står fyra funktioner till förfogande:

AUTO (n, rader)	= AUTO 20 ger efter varje RETURN ett nytt radnr, som är 20 högre än föregående rad.
RENUMBER x, y, z	= ett mycket bekvämt kommando
x	= ny radbörjan
y	= radavstånd
z	= från vilken av befintliga rader Kommandot RENUMBER 1000, 10, 100 numrerar ett basicprogram från befintlig rad nr 100 med en höjning av radnr med 10, varvid den första nya raden börjar med nr 1000. Alla språngadresser (GOTO, GOSUB, ON x GOTO/GOSUB etc) omfattas av kommandot även i de fall, där språng-

	kommandona ligger före z.
DELETE (område)	= raderar programrader i angivet område
TRON, TREFF	= kopplar in eller ut TRACE. Den vid varje tillfälle bearbetade raden visas på skärmen.

Diskettkommandon

Dessa motsvarar helt motsvarande kommandon för CBM-8000-serien.

DLOAD/DSAVE	= laddning och lagring av program
DIRECTORY	= som ordet säger så visas här adresskatalogen. Ett urval är möjligt t. ex. alla program på A.
COLLECT	= motsvarar VIC 20/64s VALIDATE
COPY	= kopierar ett program från en diskett till en annan vid dubbla disketter. Kopierar ett program inom en diskett under ett annat namn vid enkeldiskett.
RENAME	= ändrar ett registernamn.
SCRATCH	= Raderar ett register på en diskett.

De bekanta diskettkommandona från VIC 64 är också användbara. Vid felaktig status hos diskettverket får man genom variabeln DS och DS$ bild på skärmen visande felets art samt felmeddelandet. Men även basicen understödjer felet i programmen. Det finns en möjlighet att skapa fällor (eng trapping). Härför finns två kommandon:

TRAP	= fångar upp ett fel. Genom felvariablerna ER, ERR$ och EL visas de på skärmen. ER = tar tag i det senaste upptäckta felet; ERR$ = innehåller motsvarande felmeddelande i basic. EL = anger den rad i vilket felet befinner sig
RESUME	= med detta kommando fortsättes programmet utan att felet får till följd att programmet störtar. Här ges möjlighet att låta programmet fortsätta från den felaktiga raden genom att denna anges. Med RESUME NEXT fortsättes programmet direkt efter den felaktiga raden.

Dessa båda kommandon är nyttiga och besparar mycken "hacking" i onödan. Programmen blir nu störtsäkra och även om man använder STOP-tangenten, så blir det ingen BREAK utan programmet fortsätter sedan STOP släppts. Felet FILE NOT FOUND omfattas inte av rutinen.

Strukturerad programmering

Detta är programmeringens högre skola och där kommer de finesser som gör att programmen löper som smort genom minnescellerna. Även här finns det kraftfulla kommandon som tidigare saknats i VIC 64.

DO UNTIL DO WHILE EXIT LOOP UNTIL LOOP WHILE IF THEN ELSE	Med dessa kommandon är ett fullt strukturerat programförlopp möjligt. Slutvillkoret kan endera stå i slingans början (DO UNTIL/WHILE) eller i slutet (LOOP UNTIL/WHILE). Genom EXIT kan slingan lämnas.

Den nya BASIC:en 3, 5 erbjuder naturligtvis utöver vad som här meddelas även den basic som finns i VIC 20/64.

Maskinspråksmonitor TEDMON

Denna är inbyggd och förutom översättning till maskinkod så kan man också assemblera och disassemblera. TEDMON och BASIC kan löpa bredvid och utesluter ej varandra. Nu kan man enkelt göra separata assemblerprogram eller låta dem löpa som underprogram I BASIC. Påropp av TEDMON sker från BASIC med kommandot MONITOR. Här är funktionerna i maskinspråksmonitorn:

A = Assemble	omvandlar ett assemblerprogram till maskinkod.
C = Compare	jämför två sektioner i minnet och meddelar olikheterna.
D = DISSAMBLE	omvandlar maskinkod till assemblerkommandon.
F = Fill	belägger minnet med en specifiCerad byte
G = Go	startar ett assemblerprogram från en bestämd adress.
H = Hunt	avsöker minnet efter alla förekommande bytes av ett visst slag.
L = Load	laddar ett program från kassett eller diskett
M =	Memory anger innehållet i minnesceller
R = Register	meddelar värdena i 6502-register-värde.
S = Save	lagra på band eller diskett
T = Transfer	överför en minnesdel till en annan
X = Exit	lämnar TEDMON

Ytterligare kommandon

Dessa kommandon tillhör ej en bestämd kategori utan är allmänt användbara.

PRINT USING	= Definierar formatet hos en textsräng eller en talföljd vid utmatning. Framställning av tabeller blir med detta kommando ett rent nöje.
PUDEF	= Ändrar parametern hos PRINT USING-kommandot.
INSTR.	= Tillåter sammanflätning av strängar
DEC =	Omvandlar ett hextal (00 till FF) till decimaltal
HEX$	= Omvandlar ett decimaltal (0 till 65536) till ett hextal.
JOY	= Läge 1 eller 2 för Joysticks.
GET KEY	= Liknar GET-kommandot men avvaktar att tangent nedtryckes.

Jämför man VIC 64 med VIC 264 så kan man kanske säga, att den har många plussidor och att på minussidan kan upptas den sämre grafiken och de sämre ljudmöjligheterna.

Å andra sidan är BASIC:en avsevärt bättre. Med de inbyggda programmen samt maskinspråksmonitorn, så har programmerraren vunnit avsevärt. För att inte nämna den nya cursorstyrningen. Olala!

Kanske kommer VIC 264 bli det rätta mellantinget mellan hemdatorn och kontorsddatorn för små företag eller för hantverkare eller för t ex advokater, läkare och liknande vilka idag har så mycket arbete med blanketter, att det knappast blir tid över för arbete.

Sollentunamässan 25/4—28/4 1984

Det har varit microdatormässa i Sollentuna. För er som inte var där tänkte vi här återge lite av vad som fanns på mässan.

av Elisabeth Höglund



Det har varit microdatormässa i Sollentuna 25/4—28/4 1984. Som de flesta nog redan vet så fanns det en åldersgräns på mässan. Man måste ha fyllt 16 år. Att man hade satt den gränsen var för att mässan i första hand vände sig till företagare och inte till dataintresserade i allmänhet. Detta var också en av orsakerna varför VIC 20 inte fanns representerad, eftersom den i första hand är en hemdator. Spelprogrammen lyste med sin frånvaro, spel skulle helst inte förekomma. Detta var en liten förklaring till er som inte kunde besöka mässan, eller till er som saknade VIC 20 i montern.

Olika datorföretag

Det fanns otroligt många företag representerade, kända som okända. Man kan mycket väl förstå de som kände sig lite konfunderade efter en stund, för visst var det svårt att smälta alla intryck eller komma ihåg alla företag och datorer. Självklart så fanns de största och mest kända företagen representerade, som t ex IBM, Datatronic, LM Ericsson, RANK XEROX m m.

Även vi från VIC rapport var representerade. En rolig upplevelse. Vi delade ut gamla nummer av VIC rapport tillsammans med ett mässerbjudande. Dessa gick åt som smör i solsken. Redan efter första dagen var vi tvungna att ta dit fler tidningar. Kul tycker vi på redaktionen.

Det var ca 9000 personer som besökte mässan varje dag, så nog var det folk alltid. Men fredagen slog ändå rekordet. Över 10000 människor försökte klämma sig in bland montrarna.

Nyheter

Mässan bestod egentligen inte av några större nyheter, men de som fick extra mycket uppmärksamhet var pekskärmar och talande datorer. Dessa kanske man egentligen inte har någon större praktisk nytta av. I dagsläget är detta i alla fall begränsat. Men vem vet, utvecklingen går ju snabbt inom databranschen.

Att få en dator att tala är inte speciellt svårt. Man kan få en VIC 64 att tala genom den inbyggda ljudgeneratorn. Men man kan inte tala till den. TEXAS visade sin dator som talar och som man kan tala till. Problemet med denna är bara dialekten. Det kan vara svårt för en skåning eller norrlänning att tala med datorn, eftersom den är känslig för dialektala avvikelser.

Bland programvaror som väckte uppmärksamhet var Datatronics nya ord- och textbehandlingsprogram, Word Result. Frågan är om detta kommer att bli en lika stor världssuccé som Calc Result.

Det man främst kanske märkte var att det går ett IBM-syndrom genom branschen. De flesta leverantörer är angelägna om att presentera IBM-kompatibla datorer. Tyvärr är det många av de maskiner som påstås vara IBM-kompatibla som i verkligheten ej uppfyller kraven för kompatibilitet.

En av IBM:s nya datorer var den sk Peanut. Där behöver man inte någon sladdanslutning mellan datorn och tangentbordet.

En motsedd nyhet var APPLEs MC:Intoch. Ett mellanting mellan en hemdator och en mer avancerad dator. Man funderar på om dessa bägge datorer har någon meningsfull marknadsuppgift att fylla.

Det man kanske förvånades något över var att Luxor inte hade någon egen monter, utan endast företräddes av återförsäljare. Är det kanske så att NOKIA redan har slukat Luxor?

Ett nytt område som visades i ett flertal montrar var kommunikation med generella databaser. Som de flesta säkert redan vet kan man använda VIC 64 till teledata, och även den bärbara VIP 64:an.

Datatronic hade en tävling där man kunde vinna en VIP 64 och en PET 700.

De lyckliga vinnarna var Mats Norrbom, som fick med sig en VIP 64 hem, och ett ganska nystartat företag som heter Roslagens patentkonsult fick en PET 700. Ett stort grattis till er.

Intrycken från en mässa blir ändå ganska diffusa. Ni som var där vet hur mycket information man ska smälta. Men visst är det roligt att få se alla dessa otroliga apparater och dess resurser. ●

Userporten i VIC 20

I denna artikel skall jag ta upp lite grann om Userporten, hur den fungerar och hur den skall användas. Jag skall i viss mån också ta upp lite om de A/D-omvandlare som finns inbyggda i VIC-en.

av Thomas Wernersson

Men först Userporten vilken är den port som sitter längst till höger sett bakifrån. (se bild 1) Själva kontakten består utav 24 stycken pinnar. Det kan vara nog så svårt att experimentera med U-porten om man inte har någon slags kontakt eller liknande ditkopplad. En lämplig kontakt är ESM 12 DREH som finns att köpa hos ELFA för omkring 30 kronor.

Men vad kan man använda U-porten till? Ja, på den frågan finns väl en aldrig sinande ström av svar. Styrningar av robotar, lampor, m m samt mätningar av alla de slag. För att kunna ansluta yttre enheter till userporten behövs något slags interface. Det kan man, med lite färdighet i lödning och elekronik, bygga själv till en förhållandevis låg summa. Om man inte har dessa färdigheter så kan man givetvis köpa färdiga interface (anpassningsenheter). Handic har bl a ett reläkort för userporten.

För att kunna använda de åtta pinnarna C-L som ut- eller ingångar så användes två minnespositioner som styrregister. Först så anger man vilka av de åtta pinnarna som skall agera ut- respektive ingångar. Detta meddelar man datorn genom att i styreregistret 37138 ''sätta'' de bitar som skall agera utportar, och ''släcka'' de bitar som skall agera inportar.

Efter att man deklarerat vilka portar som skall kunna skicka ut signalerna så använder man sedan styrregistret 37136 för att skicka ut signalerna resepektive känna av pinnarna. Genom att ''sätta/ bitar i 37136, efter att ha deklarerat dem som utportar i 37138, så skickar man ut signaler på pinnarna. Dessa signaler är för svaga för att kunna driva reläer direkt. I stället får man konstruera en drivkrets som i sin tur, efter att ha fått signaler ur VIC-en, kan driva reläer.

Vid avläsning av pinnarna så får man släcka de bitar i 37138 som motsvarar de bitar som skall läsas av. Efter att ha släckt de bitarna så *sätts* automatiskt samma bitar i 37136. När dessa pinnar kopplas med jord så släcks respektive bitar i 37136.

Detta kan verka krångligt, så för att åskådliggöra det hela så demonstrerar jag här ett exempel tillsammans med illustrationer.

Låt oss säga att vi skall ha ut signaler på skall agera utportar till lysdioderna samt offdioder. Rad nummer 20—30 känner av pinnarna och skiftar innehållet fyra steg till vänster. När en switch är nedtryckt så släcks den biten. I vårt program så vill vi att den lysdioden skall lysa. Vi måste därför på rad nummer 40 göra en ''exklusiv-eller-operation''. D v s att satta bitar släcks och släckta bitar tänds. Sedan hoppar vi tillbaka till rad 10.

Om vi nu skulle ta och sluta switchen på bit nummer 0 skall registren se ut som i fi-

Bild 1.

Userportens placering på baksidan av VIC-en, samt dess utgångar.

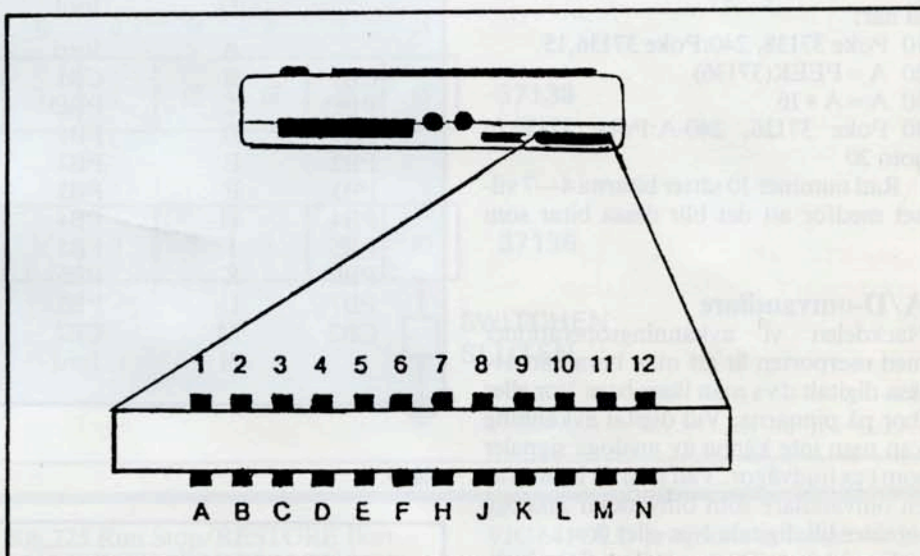
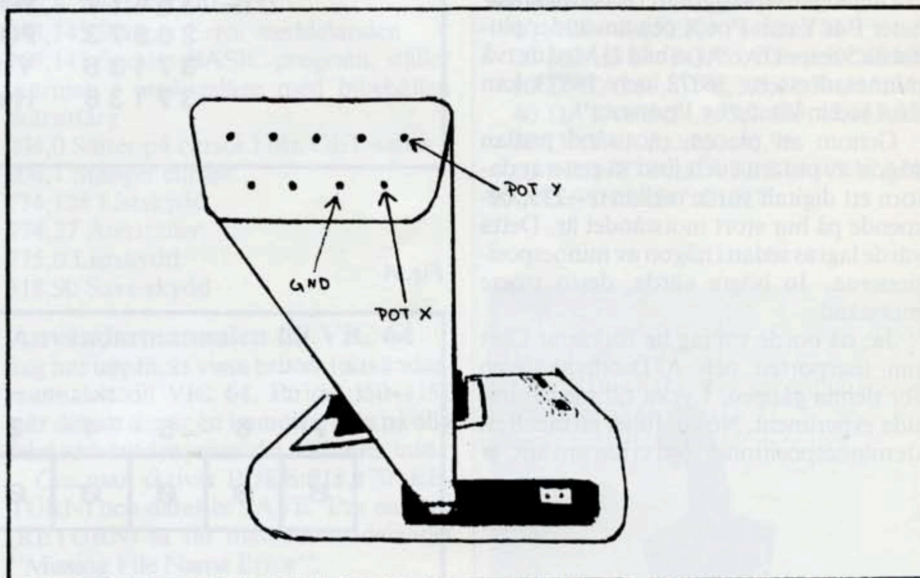


Bild 2. Placering av A/D-omvandlare i spelporten.



gur C, innan programmet hunnit tända lysdioden. Lägg märke till att bit 0 släcks när switchen sluts.

fyra pinnar och samtidigt känna av fyra andra pinnar. Det första vi gör är att starta upp VIC-en. De två kontrollregisterna ser då ut som i figur A, d v s datorn är inställd att känna av alla åtta pinnarna. Men ingen pinne är jordansluten.

Efter detta kopplar vi upp fyra lysdioder och lika många switchar enligt figur B. Om vi sedan sätter bitarna 4—7 i 37138 så har vi allting uppkopplat och klart.

Nu fattas bara ett program som känner av de fyra pinnarna och tänder respektive lysdiod. Detta program skulle kunna se ut så här:

```
10 Poke 37138, 240:Poke 37136,15
20 A = PEEK(37136)
30 A = A * 16
40 Poke 37136, 240-A:Poke 37136,0:
goto 20
```

Rad nummer 10 sätter bitarna 4—7 vilket medför att det blir dessa bitar som

A/D-omvandlare

Nackdelen vi avkänningsoperationer med userporten är att man bara kan avläsa digitalt d v s man läser bara 1:or eller 0:or på pinnarna. Vid digital avkänning kan man inte känna av analoga signaler som t ex ljudvågor. Vad som då behövs är en omvandlare som omvandlar analoga signaler till digitala 1:or eller 0:or.

En sådan A/D-omvandlare finns lyckligtvis inbyggd i VIC-en, men inte i userporten. I ställe så sitter den monterad med två ingångar i spelporten. Dessa ingångar heter Pot Y och Pot X och använder pinnarna 5 respektive 9. (se bild 2) Med de två minnesadressern 36872 och 36873 kan man sedan känna av ''potarna''.

Genom att placera motstånd mellan någon av potarna och jord så generar datorn ett digitalt värde mellan 0—255, beroende på hur stort motståndet är. Detta värde lagras sedan i någon av minnespositionerna. Ju högre värde, desto större motstånd.

Ja, då borde väl jag ha förklarat klart om userporten och A/D-omvandlaren för denna gången. Lycka till med framtida experiment. Nedan följer en tabell på de minnespositioner som vi har använt. ●

Stift hos via nr 1	Anslutning hos user port	Funktion	Adress i minnet
	1	Jord	
	2	+ 5V	
RES	3	RESET	
PA2	4	JOY Ø	37137 bit 2
PA3	5	JOY 1	37137 bit 3
PA4	6	JOY 2	37137 bit 4
PA5	7	Ljuspenna	37137 bit 5
PA6	8	Kassettswitch	37137 bit 6
	9	Serie ATN in	37137 bit 7
	10	AC 9V	
	11		
	12	Jord	
	A	Jord	
CB1	B	CB1	37148 bit 4
PBØ	C	PBØ	37136 bit Ø
PB1	D	PB1	37136 bit 1
PB2	E	PB2	37136 bit 2
PB3	F	PB3	37136 bit 3
PB4	H	PB4	37136 bit 4
PB5	J	PB5	37136 bit 5
PB6	K	PB6	37136 bit 6
PB7	L	PB7	37136 bit 7
CB2	M	CB2	37148 bit 5—7
	N	Jord	

Bild 3.

VIKTIGA MINNESPOSITIONER

36872	POT Y.
36873	POT X.
37136	Yttre Styrregistret.
37138	Huvud Styrregistret.

Fig. A

7	6	5	4	3	2	1	Ø	
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	37138
1	1	1	1	1	1	1	1	37136

Fig. B

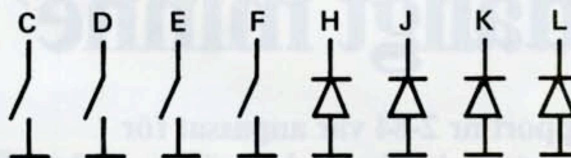
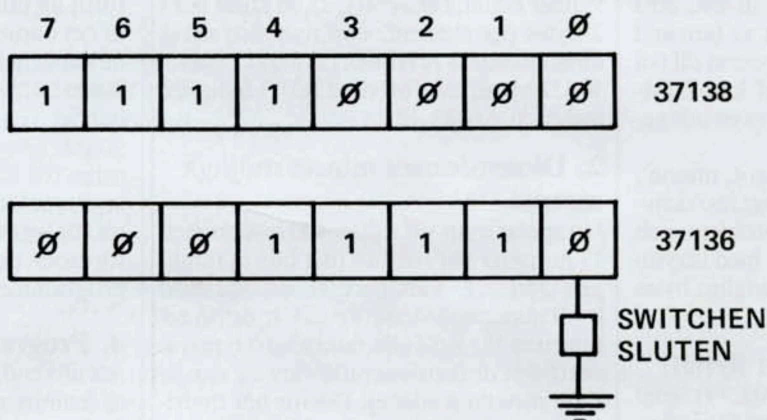


Fig. C



Joysticks användning i spel eller seriösa program

Vi har fått följande tips från Fredrik Strand. Alla hans tips är avsedda för VIC 64.

Följande behövs för att datorn ska vänta tills användaren gör ett speciellt "drag" på Joysticken.
Väntar tills användaren "drar":
Uppåt... Wait AD,1,1
Nedåt... Wait AD,2,2
Vänster... Wait AD,4,4
Höger... Wait AD,8,8
Trycker Fire... Wait AD,16,16

Istället för AD gäller följande värden:
Joystick i port nr 1: 145
Joystick i port nr 2: 56320

Användbara POKE-koder

POKE;
53272,21... Stora bokstäver
53272,23... Små bokstäver
650,255... Repetition på alla tangenter
650,0... Stänger repetition
53265,11 Stänger skärmen
53265,27 Kopplar på skärmen
808,225 Run Stop/RESTORE Bort
808,237 Run Stop/RESTORE På
788,52 Run Stop Bort
788,49 Run Stop På
646,F Textfärg X(0—15)
768,145 Stänger Error meddelanden
768,141 Suddar BASIC-program, ställer skärmen i utgångsläge med bibehållen skärmfärg
204,0 Sätter på cursor i bla GET-satser
204,1 Stänger cursor
774,128 Listskydd
774,27 Återställer
775,0 Listskydd
818,90 Save-skydd

Användarmanualen till VIC 64

Jag har upptäckt vissa brister i användarmanualen till VIC 64. På sid 150—151 står det att det är en komplett lista på alla felmeddelanden, men det stämmer inte.

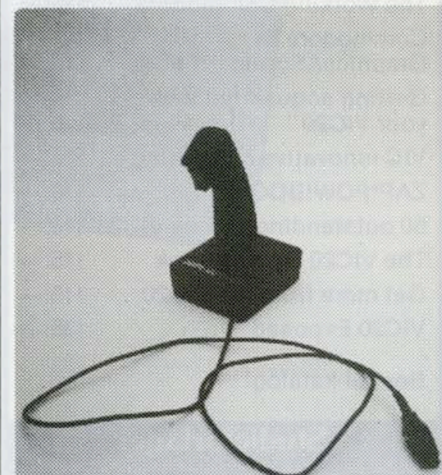
Om man skriver POKE 818,170 (RETURN) och därefter SAVE"Prg namn" (RETURN) så får man felmeddelandet "Missing File Name Error".

Om man skriver POKE 818,50 (RETURN) och därefter SAVE"Prg namn" (RETURN) så får man felmeddelandet "Illegal Device Number Error"

OLD-program

Angående ert tidigare OLD-program till VIC 64. OLD-programmet fungerade inte på min 64:a. Här följer ett fungerande OLD-program till VIC 64. Det hämtar tillbaka program som blivit suddade av NEW eller om man gjort RESET.

```
10 AD = 49152:for i = 0 to 21
20 READ D : POKEAD + I,D:NEXT
30 DATA169,8,141,2,8,32,51,165,24
40 DATA165,34,105,2,133,45,165,35
50 DATA105,0,133,46,96
60 PRINT"FÖR ATT STARTA, ANVÄND SYS"AD":CLR"
```



Gör ett ”dåligt minne” bättre

Lottoprogrammet i VIC rapport nr 2-84 var anpassat för VIC 64. Många läsare har hört av sig för att detta även skall kunna användas till VIC 20. Därför har Birger Gran omarbetatprogrammet för VIC 20.

Efter lottoprogrammet i VIC rapport nr 2-84 har jag fått brev med önskemål om hjälp att anpassa programmet till VIC 20. Att anpassa textutskrifterna till VIC 20:s radlängd torde väl alla klara av (använd förkortningar eller dela upp texten till två rader på skärmen). Däremot kan DIM-kommandot (på rad 10) behöva en utläggning.

Då 64:an har ett mycket ”gott minne’, 38911 bytes (tecken) tillgängligt läs/skrivminne (RAM)är detta inget problem, och medför gärna att man slösar med utrymmet. Men VIC 20:s 3583 tillgängliga bytes motiverar lite eftertanke.

1. Använd heltal i stället flyttal!

För flyttalsmatriser, t ex A(I, J) eller R(R), använder VIC:en 5 bytes för matrisens namn + 2 bytes för varje dimension i matrisen + 5 bytes för varje element av flyttalstyp. Om man i stället för flyttal använder heltal, t ex A%(I, J), så åtgår bara 2 bytes per element. Om matrisen alltså dimensioneras A%(400,12) åtgår ”bara” 10432 bytes, jämfört med 26071 bytes för matrisen A(400, 12)

2. Dimensionera minsta möjliga matris!

Nu spelar man väl sällan 400 system med 12 nummer per spelfält (det blir ju tämligen dyrt...). Vanligare är det väl med 8—9 nummersystem. Tyvärr är det så att matrisen tar sitt fulla minnesutrymme så snart den dimensionerats, vare sig den är fylld med data eller ej. Därför bör matrisen göras så liten som möjligt. Om man vet att inget system är större än 9 nummer/spelfält, bör andra parametern i DIM-kommandot vara 8. (Man utnyttjar index 0—8 = 9 steg.) Ex.: DIM A%(150,8) utnyttjar 2723 bytes och borde därför gå att pressa in tillsammans med programmet på en VIC 20. En viktig reservation!

3. Undvik dubbelbokföring!

Att mata in spelfälten som datasatser innebär naturligtvis att även datasatserna tar minnesutrymme (i programmet). En form av dubbel bokföring! Och detta är ju oekonomiskt med VIC 20:s begränsade kapacitet. För att slippa detta bör spelfälten i stället lagras i en datafil på kassettbandet (efter programmet). Men detta skapar genast behovet av en inmatningsrutin för spelfälten (inkl rättningsrutin, ingen matar väl in utan fel!), vidare en rutin för lagring resp hämtning av datafil — förutom de avsnitt ur det ursprungliga programmet som kollar antalet rätt.

4. Programmera minnessnålt!

Att använda låga radnummer, skriva flera kommandon på varje rad, undvika mellanslag mellan kommandon, alltid använda subrutiner när samma avsnitt ska utföras i olika avsnitt av programmet samt att undvika rem-satser, är andra knep att minska programmets minneskrav.

Nedan finner ni en utskrift av ett lottoprogram till VIC 20, med hänsyn taget till ovanstående besparingsrutiner. ●

Affärerna blommar för Commodore

För innevarande budgetår (1/7—83—30/6—84) förväntar sig Commodore nya rekordsiffror även om vissa konkurrenter försöker göra gällande att företaget inte går så bra. Under 4:e kvartalet (1/10—31/12 1983) sålde Commodore över hela världen 1.300000 datorer, varav i Västtyskland 123000 st. Under första budgethalvåret steg den totala omsättningen med 129% till 640,7 milj dollar. Enbart i den västtyska delen steg omsättningen med 309% och då var utlandsförsäljningen hos fabriken i Baunschweig ej inräknad. Julförsäljningen vilken redan den var rekordartad även här hemma verkar komma i skuggan av den orderingång man nu har. Redan har man svårigheter att leverera och värre kommer det att bli. En bidragande orsak är den mycket positiva kritik de nya hemdatorerna fått. Kanske är det bäst att beställa redan nu för att få en ny dator till jul.

BL.

```
2 DIMA%(130,8),R%(8):REM LOTTO VIC 20 * AV B.GRAN
4 PRINT"♥MENY:■":PRINT"F1 INMATN/KORR■":PRINT"F3 LAGRA ■"
6 PRINT"F5 HÄMTA DATA■":PRINT"F7 RÄTT LOTTORAD"
8 GETA$:IFA$=""THEN8
9 V=ASC(A$)-132:IFV<1ORV>4THEN8
10 ONVGOSUB20,30,40,50:GOTO4
12 FORI=1TON:GOSUB17:PRINTI":";:FORJ=0TO8
13 A=A%(I,J):IFA>0THENPRINTA;
14 NEXT:PRINT:S=S+1
15 IFS=10THENS=0:INPUT"<RETURN>";R$
16 NEXT:RETURN
17 IFS=0THENPRINT"♥RAD: INMATADE NR:■"
18 RETURN
20 S=0:GOSUB12:INPUT"<RETURN>";R$
22 PRINT"♥INMATN/KORR NR...":PRINT"(0 = TILL MENY)"
24 INPUTC:IFC=0THENRETURN
26 IFC>NTHENN=N+1:C=N
28 PRINT"RAD:";C:FORJ=0TO8:INPUTA:IFA=0THENJ=8
29 A%(C,J)=A:NEXTJ:GOTO22
30 GOSUB36:IFF$<>"J"THENRETURN
32 OPEN1,1,2:PRINT#1,N:FORI=1TON:FORJ=0TO8
34 PRINT#1,A%(I,J);CHR$(13);:NEXTJ,I:CLOSE1:RETURN
36 F$="":INPUT"♥BAND I RÄTT POS(J=JA)";F$:RETURN
40 GOSUB36:IFF$<>"J"THENRETURN
42 OPEN1,1,0:INPUT#1,N:FORI=1TON:FORJ=0TO8
44 INPUT#1,A%(I,J):NEXTJ,I:CLOSE1:RETURN
50 PRINT"♥MATA IN RÄTT RAD -":PRINT"7 ORD.NR + 2 T-NR■"
52 FORI=0TO8:INPUTR%(I):NEXT:S=0
54 FORK=1TON:R=0:T=0:FORL=0TO8:FORI=0TO6
56 IFA%(K,L)=R%(I)THENR=R+1
58 NEXTI:FORI=7TO8:IFA%(K,L)=R%(I)THENT=T+1
60 NEXTI,L:IFS=0THENPRINT"♥FÄLT ORD.RÄTT  T.NR"
62 PRINTK"      "R"      "T:IFR>4THENPRINT"* GRATULERAR! *"
64 S=S+1:IFS=10THENS=0:INPUT"<RETURN>";R$
66 NEXTK:INPUT"<RETURN>";R$:RETURN
READY.
```

Interaktiv programmering

Den här artikeln handlar om en enkel metod för att öka samspel mellan dator och användare. Det är samtidigt ett sätt att få en VIC 20 med bandspelare att kännas som en betydligt större och kraftfullare utrustning.

av Frenne Söderberg

Metoden är användbar på VIC 20, VIC 64 och andra datorer med samma sorts screen-editor för BASIC-programmet. Genom interaktiv programmering kan man använda BASIC programmet som lagringsutrymme för aktuella data. Man slipper krångla med särskilda datafiler på kassetten och det känns lättare att använda datorn till något nyttigt.

Screen Editorn

Screen-editorn är det inbyggda program som gör det enkelt att skriva eller ändra BASIC program på TV-skärmen. Du kommer så småningom att få läsa om hur man kan åka snålskjuts på screen-editorn, men först en repetition av hur den normalt används!

TV-skärmen fungerar som ditt fönster in mot datorn. Man kan också se den som din och datorns gemensamma anslagstavla för meddelanden och order. Det mesta du gör vid tangentbordet syns ju först på skärmen. Först när du trycker på return blir det datorns tur av tecknen som står i närheten av cursorn på skärmen. Du vet hur det fungerar vid programmering: från ett radnummer frammåt till max 4 rader på skärmen gäller radnumret och dator lägger texten i minnet som en programrad.

Om man glömmer RETURN och bara börjar skriva på en ny programrad så är det upplagt för trubbel. Troligtvis blir det mer än fyra rader från det första radnumret och då klipper datorn själv av. En del av programrad nummer 2 blir hängande på rad nummer 1 och resten går förlorat. Glöm därför inte att göra RETURN ordentligt efter varje sats när du nyutvecklar program. Var också extra ordentlig när du ändrat någonting i en programtext som du fått upp på skärmen med LIST-kommandot. Du måste stega med cursorn till en ofarlig position i underkanten på skärmen innan du gör något nytt kommando, t ex ett nytt LIST. Annars skadar du den programraden där cursorn just befann sig och det är inte roligt. Om du är osäker på vad raden innehöll kan du bli tvungen att gå tillbaka och ladda in kassettfilen igen, om du har en sådan.

Den här mekanismen som jag har gått igenom är viktig att förstå för fortsättningen. Lägg på minnet att ett radnummer plus text plus RETURN inom 4 skärmrader blir en programrad i datorns minne. Som sådan följer den med de andra programraderna ut på kassettbandet när man gör SAVE, även om den inte skulle ha någon direkt arbetsfunktion i programmet.

Att mötas på skärmen

Du har just fått en genomgång av de vanligaste sätten att utnyttja skärmen med LIST kommandon, cursormanöver och RETURN. Nu skall vi försöka få datorn att komma oss närmre till mötes. Den skall hjälpa oss att redigera programmen med dagsaktuella data.

Antag som exempel att du har gjort ett spelprogram som du blir gradvis skickligare i. Du vill gärna kunna anteckna ditt resultat till nästa gång och larmas om du slår ett gammalt rekord. Vi kan tänka oss att du har variabeln P för poängresultatet. Gör då så att du avslutar programmet med några nya rader ungefär så här: (vi antar radnr 500- är lediga)

```
500 PRINT"♥"
510 PR= 1650 :REM REKORD
520 IF P<=PR THEN  570
530 PRINT" GRATULERAR,":PRINT P;"=NYTT REKORD"
540 PRINT:PRINT"BEKRAEFTA RAD 510 MED"
550 PRINT"CURSOR OCH RETURN":PRINT
560 PRINT"510 PR=";P;":REM REKORD"
570 END
```

Vad händer nu? Jo programmet skriver ut:
(antag P = 2000 och att det är mer än det gamla rekordet PR)

```
GRATULERAR,
2000=NYTT REKORD

BEKRAEFTA RAD 510 MED
CURSOR OCH RETURN

510 PR=2000:REM REKORD

READY
▒
```

När du ser texten tar du bara och flyttar cursorn till raden 510 och gör RETURN. Titta sedan med LIST 510 så ser du att anteckningen snyggt och prydligt åkt på plats.

Om du någon gång längre fram presterar mer än 2000 poäng så kommer det en ny gratulation till det nya rekordet. Den här rekordbokföringen fungerar över all framtid om du så vill. Se bara till att du alltid SAVE:ar programmet till en kassett innan du stänger av datorn samt att du LOAD:ar det från samma kassett igen nästa gång du vill spela.

Med lite fantasi hittar man lätt andra tillämpningar. Jag kanske återkommer till sådana i ett senare nummer av VIC rapport och i så fall kommenterar jag gärna ideer från läsekretsen. ●

Evighetskalendern VIC 64

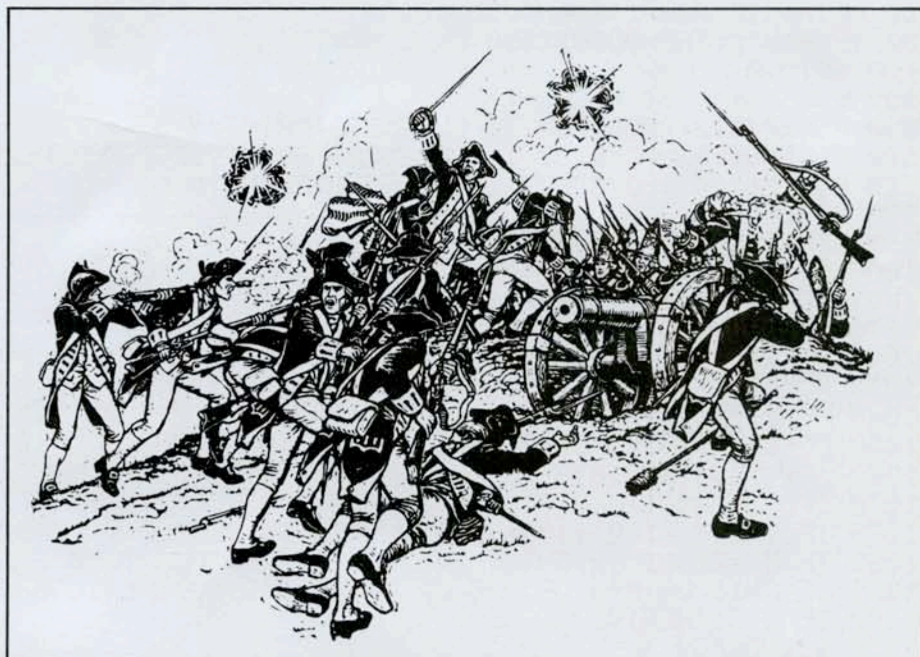
Skulle du i förskott eller i efterskott vilja ta reda på vilka veckodagar vissa händelser kommer att inträffa eller har inträffat?

I så fall skall du prova evighetskalendern.

I sig självt innehåller inte programmet några märkvärdigheter, så försök! Det kan både vara intressant och roligt.

Gustav II Adolf dog en dimmig novemberdag 1632. Närmare bestämt den 6:e. Det vet väl alla? Men har ni någon gång funderat över vilken veckodag det var? Det har jag gjort och kommit fram till att det var samma veckodag som den dag då jag fyller 100 år. Död eller levande. Den dagen är en lördag. Antalet dagar mellan Gustavs död och min 100:e födelsedag är för övrigt 149646. So what?!

Tja. .a. . Försök vara lite positiva nu är ni snälla. Själv tycker jag att det är mycket fascinerande att i förskott, eller efterskott, kunna ta reda på veckodagar då vissa historiska händelser inträffade. Som tex min 100-årsdag eller Gustavs död.



Den där dimmiga da'n i november 1632, var det en söndag? Eller kanske en måndag?

Programmet bygger på...

den Gregorianska kalendern, som lär ha införts i Sverige år 1583 och alltjämt är gällande. Där tar man hänsyn till skillnaden mellan kalenderåret och det s k tropiska året (årstidsåret). För denna tidsskillnad korrigerar man vart fjärde år, under de år som är jämt delbara med 4. Undantagen är hela århundraden som inte är jämt delbara med 400, s k sekularår. År 1900, 1800 och 1700 var alltså sekularår, men inte år 1600. Korrigeringen utförs under februari månad, som då får 29 dagar i stället för 28.

Starta programmet

När programmet startas, dyker det upp en skylt på skärmen:

1 Veckodag?

2 Dagar mellan två datum?

Vill man då ha reda på vilken veckodag 23/8 1970 var, trycker man på "1", varpå skylten "veckodag" uppenbarar sig och därunder:

Aartal? (1970)
Maanad? (8)
Dag? (23)

Datorn svarar:

Den 23/8 1970 aer en soendag

Känner man sen en längtan efter att ta reda på hur många dagar det är sedan dess, trycker man på "2". En ny skylt dyker då upp och på den står det:

"antal dagar mellan två datum"

Under denna följer så:

Aartal 1? (1970)
Maanad 1? (8)
Dag 1? (23)
Aartal 2? (1984)
Maanad 2? (3)
Dag 2? (4)
Antalet dagar aer: 4942

Observera att ovanstående parenteser endast markerar vad användaren matar in och skall inte matas in i datorn (det låter sig för övrigt inte göras).

Mina erfarenheter av programmet

I sig själv innehåller inte programmet några märkvärdigheter, med undantag av de formler som datorn använder för beräkningen. Det är sörjt för att inga dumheter skall behandlas, såsom att man försöker mata in tex den 33 augusti 1523 e dyl. Försök så får ni se vad som händer. →

BN

Kan "BN" höra av sig till VIC rapport. Du har en ersättning som väntar på dig.

Red.

```
READY.

1 PRINTCHR$(147)
5 REM **********************************
6 REM * SKYLT OCH INNEHAALSFOERKLARING *
7 REM **********************************
10 GOSUB500
20 POKE781,7 :POKE782,5 :SYS65520:PRINT"■****************************"
30 POKE781,8 :POKE782,5 :SYS65520:PRINT"■*ALMANACKAN TAECKER ALLA AAR*"
40 POKE781,9 :POKE782,5 :SYS65520:PRINT"■*FR.O.M AAR 1583.           *"
50 POKE781,10:POKE782,5 :SYS65520:PRINT"■****************************"
55 REM **************
56 REM * ALTERNATIV *
57 REM **************
60 POKE781,12:POKE782,5 :SYS65520:PRINT"■ 1 VECKODAG?                  "
70 POKE781,13:POKE782,5 :SYS65520:PRINT"■ 2 DAGAR MELLAN TVAA DATUM?   "
80 GETA$:IF A$="1"THENGOSUB1000
90 IF A$="2"THENGOSUB1500
100 GOTO80
499 END
500 POKE781,2 :POKE782,11:SYS65520:PRINT"■                 "
510 POKE781,3 :POKE782,11:SYS65520:PRINT"■EVIGHETSALMANACKA"
520 POKE781,4 :POKE782,11:SYS65520:PRINT"■                 "
530 RETURN
999 REM   ****************
1000 REM *** VECKODAG ***
1001 REM ****************
1010 PRINTCHR$(147):GOSUB500
1020 POKE781,10:POKE782,1 :SYS65520:PRINT"■** VECKODAG ** "
1030 POKE781,12:POKE782,1 :SYS65520:INPUT"AARTAL ";Y
1040 POKE781,13:POKE782,1 :SYS65520:INPUT"MAANAD ";M
1050 POKE781,14:POKE782,1 :SYS65520:INPUT"DAG    ";D
1060 IFY<1583THEN1030
1070 IFM>12ORM<1THEN1040
1075 IF(INT(Y /100)*100=Y )ANDM =2ANDD =29AND(Y /400<>INT(Y /400))THEN1050
1080 IF M =2ANDD =29AND(Y /4<>INT(Y /4))THEN 1050
1090 IF(M=11 OR M=4 OR M=6 OR M=8)AND(D>30)THEN1050
1100 IF D>31THEN1050
1110 GOSUB2000
1120 IFM<3THENF=A+D+B+C-E:GOTO1140
1130 F=A+D+B-G+H-I
1140 VD=(F+(INT(-F/7)*7))+6:IFVD=0THENVD=7
1147 REM*******************************
1148 REM**** VECKODAGSTABELL ***********
1149 REM*******************************
1150 IFVD=0THENPOKE781,20:POKE782,2:SYS65520:PRINT"DEN"D"/"M"  "Y"AER EN SOENDAG"
1160 IFVD=1THENPOKE781,20:POKE782,2:SYS65520:PRINT"DEN"D"/"M"  "Y"AER EN MAANDAG"
1170 IFVD=2THENPOKE781,20:POKE782,2:SYS65520:PRINT"DEN"D"/"M"  "Y"AER EN TISDAG"
1180 IFVD=3THENPOKE781,20:POKE782,2:SYS65520:PRINT"DEN"D"/"M"  "Y"AER EN ONSDAG"
1190 IFVD=4THENPOKE781,20:POKE782,2:SYS65520:PRINT"DEN"D"/"M"  "Y"AER EN TORSDAG"
1200 IFVD=5THENPOKE781,20:POKE782,2:SYS65520:PRINT"DEN"D"/"M"  "Y"AER EN FREDAG"
1210 IFVD=6THENPOKE781,20:POKE782,2:SYS65520:PRINT"DEN"D"/"M"  "Y"AER EN LOERDAG"
1220 IFVD=7THENPOKE781,20:POKE782,2:SYS65520:PRINT"DEN"D"/"M"  "Y"AER EN SOENDAG"
1230 RETURN
```

```
1499 REM ***************************
1500 REM *** DAGAR MELLAN DATUM ***
1501 REM ***************************
1510 PRINTCHR$(147):GOSUB500
1520 POKE781,10:POKE782,2 :SYS65520:PRINT"3** ANTAL DAGAR MELLAN TVAA DATUM **"
1530 POKE781,12:POKE782,1 :SYS65520:INPUT"AARTAL 1";Y1
1540 POKE781,13:POKE782,1 :SYS65520:INPUT"MAANAD 1";M1
1550 POKE781,14:POKE782,1 :SYS65520:INPUT"DAG 1    ";D1
1560 IFY1<1583THEN1530
1570 IFM1>12ORM1<1THEN1540
1575 IF(INT(Y1/100)*100=Y1)ANDM1=2ANDD1=29AND(Y1/400<>INT(Y1/400))THEN1550
1580 IF M1=2ANDD1=29AND(Y1/4<>INT(Y1/4))THEN 1550
1590 IF(M1=11ORM1=4ORM1=6ORM1=8)ANDD1>30THEN 1550
1600 IF D1>31THEN1550
1630 POKE781,16:POKE782,1 :SYS65520:INPUT"AARTAL 2";Y2
1640 POKE781,17:POKE782,1 :SYS65520:INPUT"MAANAD 2";M2
1650 POKE781,18:POKE782,1 :SYS65520:INPUT"DAG 2    ";D2
1660 IFY2<1583THEN1630
1670 IFM2>12ORM2<1THEN1640
1675 IF(INT(Y2/100)*100=Y2)ANDM2=2ANDD2=29AND(Y2/400<>INT(Y2/400))THEN1650
1680 IF M2=2ANDD2=29AND(Y2/4<>INT(Y2/4))THEN 1650
1690 IF(M2=11ORM2=4ORM2=6ORM2=8)ANDD2>30THEN1650
1700 IF D2>31THEN1650
1710 Y=Y1:M=M1:D=D1:GOSUB2000
1715 IFM1<3THENF1=A+D+B+C-E:GOTO1730
1720 F1=A+D+B-G+H-I
1730 Y=Y2:M=M2:D=D2:GOSUB2000
1735 IFM2<3THENF2=A+D+B+C-E:GOTO1750
1740 F2=A+D+B-G+H-I
1750 F=ABS(F1-F2)
1760 POKE781,20:POKE782,1:SYS65520:PRINT"ANTALET DAGAR AER:"F
1770 GETA$:IFA$="1"THENPRINTCHR$(147):GOTO1000
1780 IFA$="2"THENPRINTCHR$(147):GOTO1500
1790 IFA$="2"THENPRINTCHR$(147):GOTO1500
1800 GOTO1770
1999 END
2000 A=365*Y:B=31*(M-1):C=INT((Y-1)/4)
2010 E=INT(3/4*(INT(((Y-1)/100)+1))):G=INT(.4*M+2.3)
2020 H=INT(Y/4):I=INT(3/4*(INT(Y/100)+1))
2030 RETURN

READY.
```

```
100 REM *************
110 REM * INTEGRERA *
120 REM *    MED    *
130 REM * VIC20 C64 *
140 REM *           *
150 REM * FAM 84-04 *
160 REM *************
170 REM
180 REM
190 REM
200 GOTO360
210 REM
220 REM * SUBRUTIN *
230 REM
240 WD=WB-WA:WS=(FNW(WA)+FNW(WB))*WD/2:WN=1:WF=1E9
250 WD=WD/2:WG=WF:WT=0
260 FOR W=1 TO WN:WT=WT+FNW(WA+WD*(2*W-1)):NEXT
270 WT=2*WD*WT:WF=(WS+2*WT)/3:WS=(WS+WT)/2:WN=2*WN
280 PRINT"WAIT"WN+1;WF:IF ABS(1-WG/WF)>10↑(-WE) GOTO 250
290 RETURN
300 REM
310 REM * EXEMPEL
320 REM * TREDJEGRADS
330 REM * FUNKTION
340 REM * EXAKT SVAR
350 REM *     =1
360 REM
370 PRINT"■TREDJEGRADSFUNKTION
380 WA=0:WB=1:WE=5
390 DEF FNW(X)=1-2*X+6*X*X-4*X*X*X:GOSUB 240
400 PRINT"SVAR="WF
410 REM
420 REM * SINUSFUNKT.
430 REM * EXAKT SVAR
440 REM *     =1
450 REM
460 PRINT"■SINUSFUNKTION
470 DEF FNW(X)=SIN(X)
480 WA=0:WB=π/2:WE=4:GOSUB 240
490 PRINT"SVAR="WF
READY.
tredjegradsfunktion
WAIT 3   1
WAIT 5   1
SVAR= 1
sinusfunktion
WAIT 3   1.00227988
WAIT 5   1.00013459
WAIT 9   1.00000829
WAIT 17   1.00000052
SVAR= 1.00000052

READY.
```

→

Nedan publiceras två artiklar som visar hur du kan utnyttja din VIC för att lösa avancerade matematiska problem.

Räkna med VIC!

Många problem löses genom beräkning av en eller flera integraler. Numerisk beräkning av en integral gör du snabbt och enkelt med din hemdator. Går du på gymnasiet eller högskola kan du nu kontrollera om ett analytiskt beräknat svar är rätt eller ej. Den här presenterade integrations-algoritmen är iterativ och avbryter sig när tillräckligt god noggrannhet har erhållits.

Gör så här. Mata in programmet nedan. Definiera din egen funktion, integrationsgränser och önskad noggrannhet. Sen anropas subrutinen för beräkning av integralen och svaret erhålles. Följande beteckningar används:

FNW(W) = funktionen att integrera
WA,WB = undre och övre integrationsgränser
WE = antalet korrekta siffror i svaret
WF = integralens värde (svaret)

Som framgår av exemplet nedan erhålles exakt svar för polynom med gradtalet mindre än fyra. Sinusintegralen kräver 17 st funktionsberäkningar och tar cirka två sekunder i beräkningstid. Observera den goda noggrannheten. Långsammare konvergens erhålles till exempel för funktioner med oändlig derivata i någon punkt. Beräkna till exempel integralen av SQR(x) mellan 0 och 1. ●

Matematikfunktioner på VIC

av Toni Roth

Följande artikel beskriver hur man kan utnyttja sin VIC till att lösa svårare ekvationer. Artikeln är i första hand avsedd för dig med grundläggande kunskaper i matematikens värld. Men men kanske den även kan hjälpa dig med mindre djupa kunskaper att på ett roligare och enklare sätt klara av din matteläxa.

Iterationer

Det finns två viktiga begrepp inom matematiken: analytisk och numerisk matematik. Den analytiska delen behandlar exakt matematik: Ta till exempel ekvationen xü2—x = x*(x—1) = = > x1 = 0, x2 = 1 (''ü'' betyder ''upphöjt till'').

I den numeriska delen löser man inte ekvationerna. Om ekvationerna kan lösas exakt utan allför stort besvär, ska man naturligtvis hellre göra det. Men om ekvationen är olöslig, så måste man ta till andra metoder. För att åskådliggöra detta kan vi ta ekvationen xü5 — 3*x = 17. Denna ekvation är inte olöslig, men är väldigt jobbig att lösa exakt. Ett sätt att lösa denna ekvation numeriskt är att pröva sig fram med olika värden på x tills man får fram ett värde som stämmer tillräckligt bra. Exempel: x = 0 = = > 0ü5 — 3*0 = 0, för litet; x = 2 = = > 2ü5 — 3*2 = 26, för stort osv...

Detta är ganska jobbigt och tidsödande, så det vore bättre om man kunde formulera en algoritm för prövandet och sedan låta en dator ta hand om jobbet. Det finns flera välprövade algoritmer, varav jag tänker ta upp de tre vanligaste.
Det som är gemensamt för de tre metoderna är att man först skriver om ekvationen så man får 0 på ena sidan likhetstecknet. I vårt fall: xü5 — 3*x = 17 = = > xü5 — 3*x — 17 = 0. Vi ska alltså lösa f(x) = xü5 — 3*x — 17.

Den enklaste metoden kallas intervallhalveringsmetoden och går i stort sett ut på metodiskt prövande.

Intevallhalveringsmetoden

Man börjar med att välja (pröva) ut två gränser som ligger på varsin sida om nollstället. Här till exempel 0 och 2. Man tittar på funktionsvärdet mittemellan dessa två gränser.: f(1) = 1ü5 — 3*1 — 17 = —19 < 0. Både funktionsvärdet för för vänstra gränsen och för mittgränsen är negativa. Det betyder att nollstället måste ligga mellan mittgränsen och högra gränsen. Detta ser man lättare om man ritar en figur:

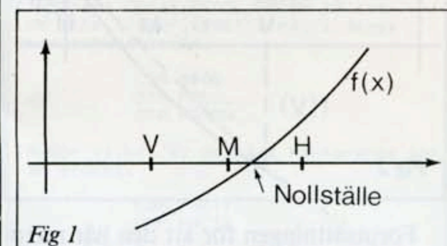


Fig 1

Vi ser att om funktionsvärdet för vänstergränsen och mittgränsen ligger på samma sida om 0, så ligger nollstället mellan mittgränsen och högergränsen. Men om f(v.g) och f(m.g) har samma tecken, så blir f(v.g)*f(m.g) > 0 (v.g = vänstergränsen, m.g = mittgränsen...) Dvs: om produkten >0 så ligger nollstället mellan m.g och h.g, annars mellan v.g och m.g.

Nu har vi två nya gränser, 1 och 2. Vi kan nu göra om samma sak en gång till och en gång till..... Mittgränsen kommer att anta värdena 1, 1.5, 1.75, 1.875, 1.8125 ..... Med andra ord: Vi kommer närmare och närmare nollstället, vi itererar.
När ska man då sluta att iterera? Man kan sluta när vänstergränsen ligger tillräckligt nära högergränsen. Ett bättre sätt är att sluta när f(m.g) är tillräckligt nära 0. För ett litet fel i x leder ofta till ett större fel i f(x). Om vi kallar vänstergränsen för V, högergränsen gör H och mittgränsen för →

M, så kan vi skriva ett litet program som hittar lösningen till f(x) = 0:

```
10 V=0 : H=2 :
DEFFNF(X)=Xü5—3*X—17
20 M=(H+V)/2
30 IF FNF(V)*FNF(M)>0 THEN V=M
40 IF FNF(V)*FNF(M)<0 THEN H=M
50 IF FNF(M)>1E—6 THEN GOTO 20:
REM Sluta när f(M) nära 0
60 PRINT "Lösning: x=";M
70 END
```

Detta program kommer naturligtvis komma fram till samma resultat varje gång, men ändrar vi i rad 10, så kan vi lösa andra ekvationer.

Det kan krävas rätt så många iterationer innan intervallhalveringsmetoden ger resultat. Det finns andra metoder, vilka som regel är snabbare. En av dessa är sekantmetoden.

Sekantmetoden

Precis som i förra metoden väljer man två gränser V och L. Men istället för att räkna ut M som (V + H)/2, så tar man M = H — f(H)*(H—L)/(f(H)—f(L)).

Vad är nu detta?! Om vi ritar en figur är det lättare att förklara.

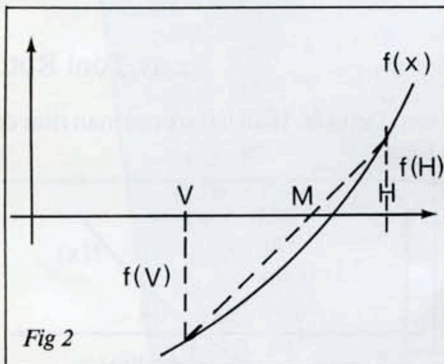


Fig 2

Förutsättningen för att den här metoden ska vara snabbare är att f(x) går någerlunda rakt från V till H. I så fall kommer den räta linjen mellan punkterna (V,f(V)) och (H,f(H)) att skära x-axeln nära nollstället. Om vi utgår från H, så kommer f(H) och lutningen på den räta linjen tala om hur man räknar ut M:

lutningen = (f(H)-f(V))/(H-L) (f(V) är negativ) men

lutningen = f(H)/(H-M) gäller också = = >

$$\frac{f(H)—f(V)}{H—L} = \frac{f(H)}{H—M} ==> H—M = f(H)^* \frac{H—L}{f(H)—f(V)} ==> M = H—f(H)^* \frac{H—L}{f(H)—f(V)}$$

Sedan kontrollerar man f(V)*f(M) och byter gränser precis som i intervallhalveringsmetoden.

Det finns en metod som är ännu bättre än de båda andra, nämligen Newton-Raphsons metod. Denna kräver dock lite mer förarbete.

Newton-Raphsons metod

Man väljer ett startvärde x0, vilket som helst nästan. Sedan beräknar man ett bättre värde med formeln

$$x1 = x0 — \frac{f(x0)}{f'(x0)}$$

Där f(x0) är derivatan av f(x) i punkten x0.

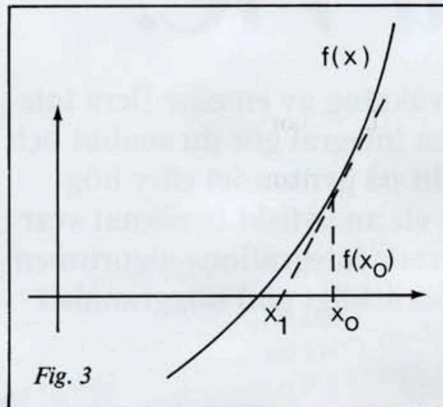


Fig. 3

$$==> f'(x0) \quad \frac{f(x0)}{x0—x1} \quad ==>$$

$$x0—x1 = \frac{f(x0)}{f'(x0)} \quad ==>$$

$$x1 = x0— \frac{f(x0)}{f'(x0)}$$

Denna metod är oftast avsevärt mycket snabbare än båda föregående, men till skillnad från dessa, kan denna spåra ur, hamna i en oändlig slinga.

Vi har en punkt x0. Fför att få ett bättre värde drar vi tangenten genom punkten (x0,f(x0)). Tangenten har lutningen f'(x0). Om f(x) inte svänger för mycket kommer tangenten att skära x-axeln nära nollstället. Denna punkt kallar vi x1. Då gäller:

lutningen = f'(x0)
lutningen = f(x0)/(x0-x1)

Om vi startar i x0, så kommer derivatan leda oss till x1. Därifrån leder derivatan oss till x2, men x2 = x0. Vi har inte kommit närmare nollstället utan kan snurra runt på samma sätt hur länge som helst. Man kan dessutom hamna i en punkt x där f'(x) = 0, och dela med 0 går ju inte. Ytterligare ett problem är att vi måste räkna ut f'(x), vilket kan vara nog så svårt ibland. I de flesta fall ställer allt detta inte till några större problem. Så Newton-

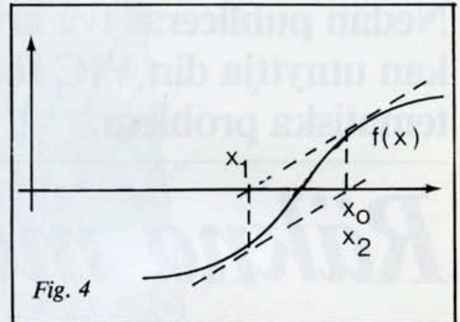


Fig. 4

Raphsons metod är för det mesta den bästa metoden. För fullständighetens skull tar jag med ett program som löser ekvationen xü5—3*x—17 = 0 enligt den sista metoden:

```
10 X=2
20 DEFFNF(X)=Xü5—3*X—17
30 DEFFNFD(X)=5*Xü4—3
40 X=X—FNF(X)/FNFD(X)
50 IF FNF(X)>1E—6 THEN GOTO 40
60 PRINT "Lösning: x=";X
70 END
```

Som en jämförelse mellan de olika metoderna kan jag tala om att vid testkörning krävdes det 25 iterationer för intevallhalveringsmetoden innan den kom fram till svaret (1.86555868), 10 för sekantmetoden samt 4 för Newton-Raphsons metod. Dessa värden varierar dock mycket om man tar andra startvärden.

Iterationer kommer till användning inom alla möjliga områden. Ta till exempel de som sysslar men konstruktionsberäkningar på broar och dylikt. De stöter på mer eller mindre olösliga problem ganska ofta, och då spelar iterationsmetoden en ovärderlig roll. Jag hoppas att du som läst ända hit inte tycker att det var alltför obegripligt, och att du fått ut något som du eventuellt kan använda. Om inte annat har du fått lite övning i matte, och det kan alltid vara användbart. ●

Multiplikation på 6502'an

Hur utför man en multiplikation om man måste göra den i assembler eller maskinkod. I denna artikel skall vi försöka reda ut begreppen en aning och förhoppningsvis ge några värdefulla tips.

De rutiner som redovisas här kommer att vara skrivna som subrutiner vilket gör det mycket lämpligt för dig att spara dessa sidor då de senare kan vara en värdefull referens. Detta gäller annars allmänt. En duktig programmerare löser aldrig en uppgift som redan tidigare är löst.

En multiplikation är ju tyvärr något som det inte finns någon maskinkodsinstruktion för och alltså är det upp till programmeraren att skapa denna rutin. Att den behövs någon gång i programmeringsarbetet det är något som man kan vara helt säker på.

Innan man gör ett program så är det dock MYCKET VIKTIGT!!! att man sätter sig in i uppgiften så låt oss först studera hur en multiplikation går till:

Vi kan direkt konstatera att det finns två sätt att utföra en multiplikation på. Antag t ex att vi skall utföra multiplikationen

8 x 12 = ?

För den som har multiplikationstabellen i färskt minne blir detta inget större problem, 96 dyker upp i huvudet alldeles av sig själv. För oss som "vilade" oss på mattetimmarna så krävs det lite räknearbete. Vi ställer upp:

```
   12
 x 8 1
 -----
   96
```

och får på så sätt det rätta svaret. Vad är det nu som vi gör här? 8x16 betyder helt enkelt att 16 skall adderas till sig själv åtta ggr d v s

16+16+16+16+16+16+16+16=96

eller

8+8+8+8+8+8+8+8+8+8+8
+8+8+8+8+8=96

På samma sätt så skulle
2x1034=2068
kunna skrivas
1034+1034=2068
eller också kunde man addera två ettusentrettiofyra ggr (pust fräs).

Att koda detta skulle inte vara särskilt svårt. Vi tar först och jämför de två faktorerna för att ta reda på vilken av dem som är minst och använder denna som en loopräknare som talar om hur många ggr vi skall addera den andra faktorn med sig själv för att få resultatet. Koden för ett sådant program skulle se ut något i den här stilen.

```
*************************************
*        MULTIPLIKATION (MULTS)     *
*             *
* succesiva metoden                 *
*                                   *
* BESKRIVNING:                      *
*                                   *
* Två 8-bits tal multipliceras och  *
* ger ett 16-bits resultat.         *
*                                   *
*                                   *
* STARTVILLKOR:                     *
*                                   *
* FAKT1 innehåller faktor 1.        *
* FAKT2 innehåller faktor 1.        *
*                                   *
*                                   *
* SLUTVILLKOR:                      *
*                                   *
* PRODL innehåller lågbytet.        *
* PRODH innehåller högbytet.        *
* Dvs                               *
* PRODUKT = FAKTOR 1 * FAKTOR 2     *
*                                   *
*                                   *
*                                   *
*************************************
```

```
        ORG  'startadress'

FACT1        EQU  $FC       ;faktor 1
FACT2        EQU  $FD       ;faktor 2
PRODL        EQU  $FE       ;LSB av produkt
PRODH        EQU  $FF       ;MSB av produkt

;Initiera först PRODL/PRODH så att
;de inte ligger något skräp i dessa
;minnespositioner.

             LDA #$00
             STA PRODL
             STA PRODH

;Avgör vilken av de båda faktorerna som
;är störst.

             LDA FACT1
             CMP FACT2

;CARRY sätts av denna del om

;FAKTOR1 >= FAKTOR2

             BCS SWAP
;Här ligger dom som dom skall, dvs
;FAKTOR2 > FAKTOR1

             BCC CONT  ;Utför mult

;Byt plats på faktorerna.

SWAP         PHA
             LDA FAKT2
             STA FAKT1
             PLA
             STA FAKT2

;MULTIPLICERA DE TVÅ FAKTORERNA

             LDX FAKT2 ;loop räknare
             INX
START        DEX       ;KLAR?
             BEQ KLART ;JA!
             CLC
             LDA FAKT1
             ADC PRODL ;Addera FAKTOR1
             STA PRODL ;till delprodukt.
             BCC START ;OBS! 16-BITS
             INC PRODH ;addition
             JMP START
             RTS

             END
```

→

Se hit!

Du som sänder in programförslag till redaktionen. Det skulle vara bra om du skickade med band eller en diskkopia på programmet. Självklart skickar vi tillbaka dessa.

Vårt arbete skulle på detta sätt underlättas väsentligt, och garantera ett bättre tryck.

Red.

→ Trots att vi här ansträngde oss för att få så få loopar som möjligt så har den här metoden enorma nackdelar. Tänk dig t ex ett fall där
256x256
skall multipliceras. Vi skulle då behöva gå igenom loopen 256 ggr. Låt oss istället titta på en annan metod. För att kunna göra detta så måste vi försöka förstå vad som händer när vi utför en handmultiplikation. Låt oss därför utföra en sådan multiplikation.

```
    234
  x  22
  -----
    468
  + 468
  -----
   5148
```

Frågan är nu om det går att utföra en liknande multiplikation binärt. Låt oss testa detta:

```
234 = %11101010

22  = %00010110

        11101010
    *   00010110
    ------------
        00000000
       11101010
      11101010
     00000000
    11101010
   00000000
  00000000
+ 00000000
----------------
 0001010000011100  = 5148
```

Tittar man på ovanstående och har skift instruktionerna på 6502'an i minnet så kommer man genast på ett sätt att lösa problemet på ett betydligt smartare sätt än det vi tidigare använde oss av. D v s det enda vi behöver göra är att skapa en loop som skiftar den ena faktorn ett steg åt vänster på varje varv och adderar resultatet till delprodukten om den andra faktorn har en etta i denna position. I annat fall tar vi bara nästa skiftning o s v. Detta medför att vi alltid klarar oss med 8 varv i loopen och metoden blir följdaktligen betydligt snabbare än den succesiva metoden. Här kommer ett program som utför en multiplikation enligt denna metod:

```
*************************************
*         MULTIPLIKATION (MULTB)    *
*              *
* Binära metoden                    *
*                                   *
* BESKRIVNING:                      *
*                                   *
* Två 8-bits tal multipliceras och  *
* ger ett 16-bits resultat.         *
*                                   *
*                                   *
* STARTVILLKOR:                     *
*                                   *
* FAKT1 innehåller faktor 1.        *
* FAKT2 innehåller faktor 1.        *
*                                   *
*                                   *
* SLUTVILLKOR:                      *
*                                   *
* PRODL innehåller lågbytet.        *
* PRODH innehåller högbytet.        *
* Dvs                               *
* PRODUKT = FAKTOR 1 * FAKTOR 2     *
*                                   *
*                                   *
*                                   *
*************************************
```

OBS! att FAKTOR1 består av två bytes extrabytet, d v s högbytet behövs för vi skall kunna skifta denna mot höger.

```
MULTB     CLC
          LDA #$00
          STA PRODL
          STA PRODH
          STA FAKTOR1+1
;
;minnen för resultat och faktor1:s
;'skiftbyte' initierade.
          LDX #$07     ;8 varv i loop
;
START     LSR FAKT2
          BCC SKIFT    ;ingen add,hoppa
          CLC          ;addera delprod
          LDA PRODL
          ADC FAKT1
          STA PRODL
          LDA PRODH
          ADC FAKT1+1
          STA PRODH
SKIFT     ASL FAKT1
          ROL FAKT1+1
          DEX
          BPL START    ;nästa varv
          RTS
```

Om du känner dig osäker på vad som händer i exemplet ovan så ta och testa genom att köra rutinen tillsammans med papper och penna. Detta är ett mycket bra sätt att lära sig programmering och man får in metoder och funktioner i minnet mycket snabbt, även om detta sätt att lära sig programmering kan vara väl så jobbig.

För fullkomlighetens skull så visar vi även en rutin som utför en division mellan en 16-bits divisor och en 8-bitars dividend. Metoden som används är analog med den i vår senaste multiplikations rutin, d v s vi använder oss av både skift och subtraktion. Vi kommenterar inte denna rutin utan lämnar detta åt dig som övning:

```
*************************************
*          DIVISION (DIVB)          *
*                    *
* skift/subtr-metod                 *
*                                   *
* BESKRIVNING:                      *
*                                   *
* Ett 16-bits tal divideras med ett *
* 8-bits tal och lämnar kvoten som  *
* ett 16-bits tal samt resten som   *
* ett 8-bits tal.                   *
*                                   *
* STARTVILLKOR:                     *
*                                   *
* DVSOR  innehåller tal att dela    *
*        med dvs divisor, 8-bitar   *
* DVDNDL Lågbyte av dividend.       *
* DVDNDH Högbyte av dividend.       *
*                                   *
*                                   *
* SLUTVILLKOR:                      *
*                                   *
* KVOTL  Lågbyte av resultat.       *
* KVOTH  Högbyte av resultat.       *
* REST   REST för resultat.         *
* dvs                               *
* KVOT=DIVIDEND/DIVISOR             *
* Två 8-bits tal multipliceras och  *
* ger ett 16-bits resultat.         *
*                                   *
*                                   *
*************************************
```

Multiplikation i assembler.

```
          ORG 'programstart'
;
DVSOR     EQU $E0    ;Divisor
DVDNDL    EQU $E1    ;Dividend lågbyte
DVDNDH    EQU $E2    ;Dividend högbyte
DVDNDS    EQU $E3    ;Dividend skiftb.
KVOTL     EQU $E4    ;Kvot lågbyte
KVOTH     EQU $E5    ;Kvot högbyte
REST      EQU $E6    ;Rest av division
;
;
;Initiera minnespositioner
;
DIVB      LDX #$00
          STX DVDNDS ;nollst. skiftbyt
          LDA DVSOR
;
IGEN      BMI START  ;inget skift!
          INX        ;ökA skifträknare
          ASL DVDNDL ;skifta dividend
          ROL DVDNDH
          ROL DVDNDS ;in i skiftbyte
          ASL DVSOR  ;skifta divisor
          JMP IGEN   ;kolla nästa MSB
;
START     LDY #$0F   ;16 varv i loop
;
NäSTA     ASL DVDNDL ;skifta dividend
          ROL DVDNDH
          ROL DVDNDS
          BCS CARRY2 ;C=1 --> subtr.
          LDA DVDNDS
          CMP DVSOR  ;DVSOR < DVDNDS ?
          BCS CARRY2 ;nej!.>,-->subtr
          ASL KVOTL  ;skifta 0 in i
          ROL KVOTH  ;kvot
          JMP FORTS
;
CARRY2    SEC
          ROL KVOTL  ;skifta 1 in i
          ROL KVOTH  ;kvot
          SEC
          LDA DVDNDS
          SBC DVSOR
          STA DVDNDS
;
FORTS     DEY
          BPL NäSTA  ;en gång till
;
SKIFT     DEX
          BMI KLART
          LSR DVDNDS
          JMP SKIFT
;
KLART     STA REST
          RTS
```

Bygga en mikrodator och lite assembler.

I den här artikeln skall vi först och främst ta en titt på några assemblerlösningar och sedan ge några tips till dig som har planer på att bygga en egen mikrodator.

Låt oss först titta på ett program som kan användas för att flytta ett block i minnet till en annan area. Detta är något som man ofta råkar ut för och det kan därför vara nyttigt att titta på en sådan rutin. Här är ett sätt att göra detta på:

Blockförflyttning (BLKMOV)

```
BESKRIVNING:

Ett block av valfri storlek
flyttas till en anan area.
Flyttningen börjar från den lägsta
adressen dvs tar ej hänsyn till
överlappning.

STARTVILLKOR:

START startadress för block att
      flytta.
ANTAL Det antal bytes som skall
      flyttas.
DEST  Ny startadress för blocket.

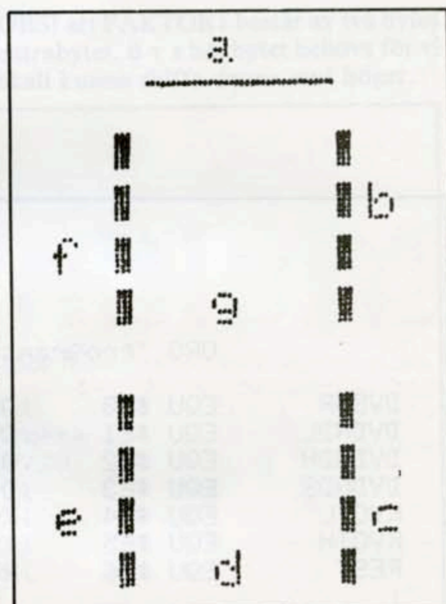
SLUTVILLKOR:

          ORG  'startadress'
BLKMOV    LDY #$00   ;nolla index
          BEQ DELSID ;inga sidor
          LDX ANTAL+1;antal sidor
;
;Här flyttas en hel sida i taget
;
SIDA      LDA (START),Y ;flytta ett
          STA (DEST), Y ;byte
          INY
          BNE SIDA      ;nästa byte
          INC START+1
          INC DESTA1
          DEX
          BNE SIDA      ;nästa sida
;
;
;här flyttas en del av sida, dvs det
;återstår sedan alla hela sidor
;har flyttats
;
DELSIDA   LDX ANTAL
          BEQ KLART
LOOP      LDA (START),Y
          STA (DEST ),Y
          INY
          DEX
          BNE LOOP
;
;
KLART     RTS
;
;
```

Denna rutin är mycket enkel att ändra så att den passar även i ett fall när man har minnesblock som överlappar varandra. Man måste då ha två rutiner. En som ovan och en annan som startar flytten från den övre delen av blocket och i inledningen av rutinen avgöra vilken av dessa som skall användas.

Vår andra assemblerlösning, och sista, är en rutin som kanske kan vara användbar för den som tänkt bygga ett "litet" experimentsystem. Rutinen omvandlar ett BCD tal i akumulatorn till den motsvarande "7-segmentskoden". Detta är ju alltid det enklaste och billigaste sättet att presentera sin data på om man nu inte är riktigt ambitiös och satsar på en bildskärmsorienterad enhet från början. För att kunna koda alla tal mellan 0-9 krävs att vi har 4 bitar. Dessa kan alltså i realiteten koda ett valfritt hextal ($0-$F). För att driva en 7-segmentsdisplay så kan vi välja tex 74LS47 som har utgångar som ligger aktiva låga, dvs vi har en enhet som driver en gemensam anod LED. I fig 1 kan Du se hur man skulle kunna koppla en enkel display. Här antar vi dock att vi inte har tillgång till denna krets utan bara en drivare (fig 1b).

På vår drivare har vi utgångar från a-g som var och en skall kopplas till ett segment på LED:en. Vi bestämmer oss för att koppla dem så här:



Dvs den kod som vi skall lägga på ingångarna skall vara:

visning DECIMALT	Kod till driver HEX	A	B	C	D	E	F	G
0	01	0	0	0	0	0	0	1
1	4F	1	0	0	1	1	1	1
2	12	0	0	1	0	0	1	0
3	06	0	0	0	0	1	1	0
4	4C	1	0	0	1	1	0	0
5	24	0	1	0	0	1	0	0
6	60	1	1	0	0	0	0	0
7	0F	0	0	0	1	1	1	1
8	00	0	0	0	0	0	0	0
9	0C	0	0	0	1	1	0	0

Där en nolla på ingången (tex A) medför att motsvarande segment på utgången skall vara tänt (tex a).

Hur skall vi nu koda detta?

En metod skulle vara att testa efter hand som programmet går framåt och sedan hoppa ur när en match uppstod. Denna teknik är dock både klumpig och utrymmeskrävande. Bättre är då att lagra de giltiga koderna i en tabell och sedan i en loop jämföra akumulatorernas innehåll med elementen i denna tabell i tur och ordning. Så här skulle ett progra enligt denna teknik kunna se ut: →

BCD – – –> 7-SEGMENT (BCD->7)

```
BESKRIVNING:

Ett BCD kodat tal i akumulatorn
omvandlas till motsvarande i
7-segmentskod för en gemensam
anod LED.OBS! att en gemensam
katod bara erfodras att kompli-
mentet tas av resultatet.Om A
inte innehöll ett BCD tal så
retuneras $FF i A.

STARTVILLKOR

Akumulatorn innehåller ett
BCD kodat tal.

SLUTVILLKOR:

Akumulatorn innehåller ett
7-segmentkodat tal eller $FF om
akumulatorn inte innehöll ett BCD
kodat tal.

;
SEGMNT      BYT $01,$4F,$12
            BYT $06,$4C,$24
            BYT $60.$0F.$00
            BYT $0C
;
;
BCD->7      CMP #$0A     ;BCD KOD?
            BCC OMV      ;NEJ!
            JMP KLART
;Här om giltig BCD kod.
;
            TAX          ;Pekar
            TAY          ;loopräknare
            INY
LOOP        LDA SEGMNT,X;ta in kod
            DEY
            BNE LOOP
;
KLART       RTS
```

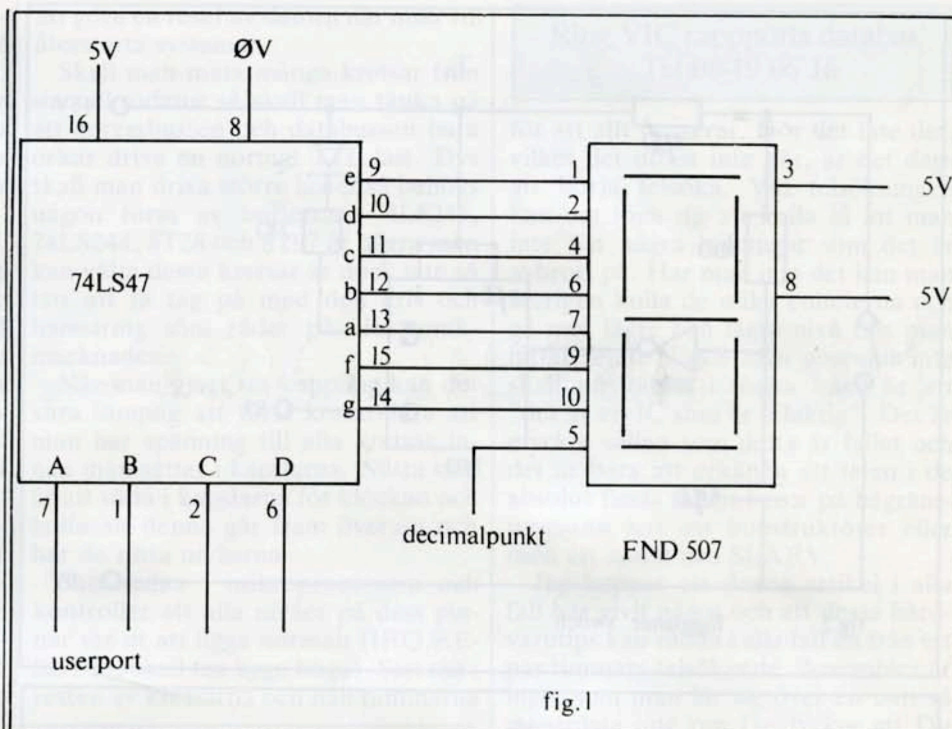


fig.1

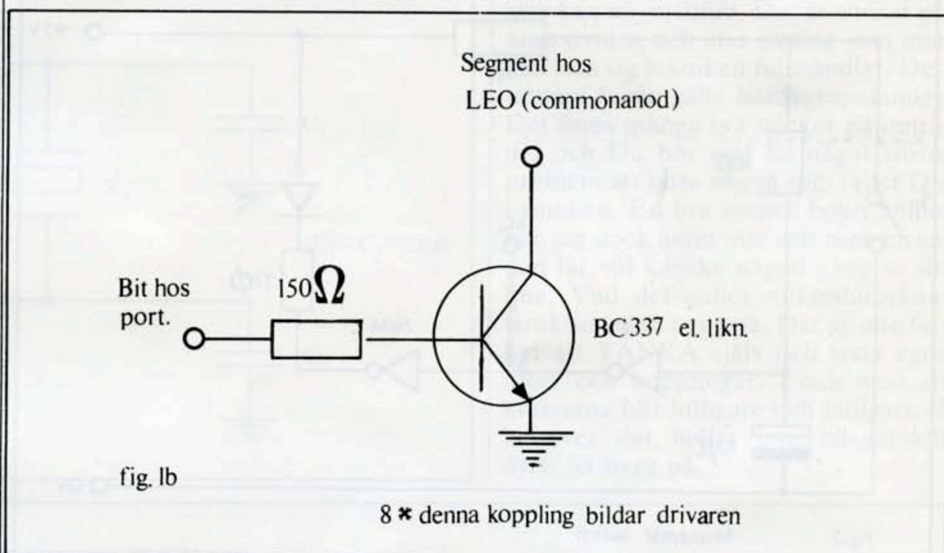


fig. lb

Lite hårdvarutips för hemmaplan

Anledningen till att man någon gång lägger ned möda på att lära sig assemblerspråket är ju ofta så enkel som att man måste kunna styra ett hemmabygge. Det är ju trots allt så att om man bygger själv så måste man också göra alla styrrutiner själv. I ett kommerciellt system som CBM maskinerna, där man kanske har tillgång till ett högnivåspråk, är det knappast lönt att hålla på med assembler då högnivåspråket klarar av att lösa de flesta uppgifter. Det är bara när man vill ha full kontroll över sitt system och extrem snabbhet som tillgripandet av assemblerspråket är berättigat.

Det första man skall tänka på när man skall bygga sig ett experimentsystem är att se till att man gör det lätt för sig. Att tex försöka plocka ihop något på en veroboard är i dessa sammanhang dömt att misslyckas. Bättre är då att köpa sig en virpenna med tillhörande virsocklar och experimentkort. Det kostar någon hundralapp att köpa dessa prylar men man finner snart att det är en MYCKET god investering. Metoden "JUST WRAP" som bl.a. säljs av ELFA är den jag själv använder mig av och klart rekommenderar. Vid denna metod så kan man lägga hur långa kontinuerliga kedjor som helst s.k. "daisy chain", dessutom så slipper man helt det mycket tidsödande momentet att skala tråd. Många andra virmetoder finns och har Du chansen så testa innan Du bestämmer Dig för köp.

En annan bra metod är att använda sig av en kopplingsplatta. Samma sak gäller även här, som vid virning, att man får satsa några hundralappar men också i detta fall är de välinvesterade. ELFA och i stort sett alla andra lagerför en kopplingsbräda som kallas "BIMBOARD" och är av mycket hög kvalitet. De är dessutom lätta att bygga ut till större enheter som blir både stabila och lätthanterliga.

Själv brukar jag börja med att koppla upp enskilda enheter, tex en klocka på kopplingsbräda för att se om uppkopplingen fungerar som jag tänkt mig. När detta är verifierat så åker virpennan fram och i detta moment har man chansen att se hur den enskilda kopplingen fungerar tillsammans med de andra delarna. Fungerar dessa här så vet man i alla fall att de enskilda delarna fungerar och felsökningen underlättas avsevärt, samtidigt som virmetoden gör det lätt att modifiera kopplingen. →

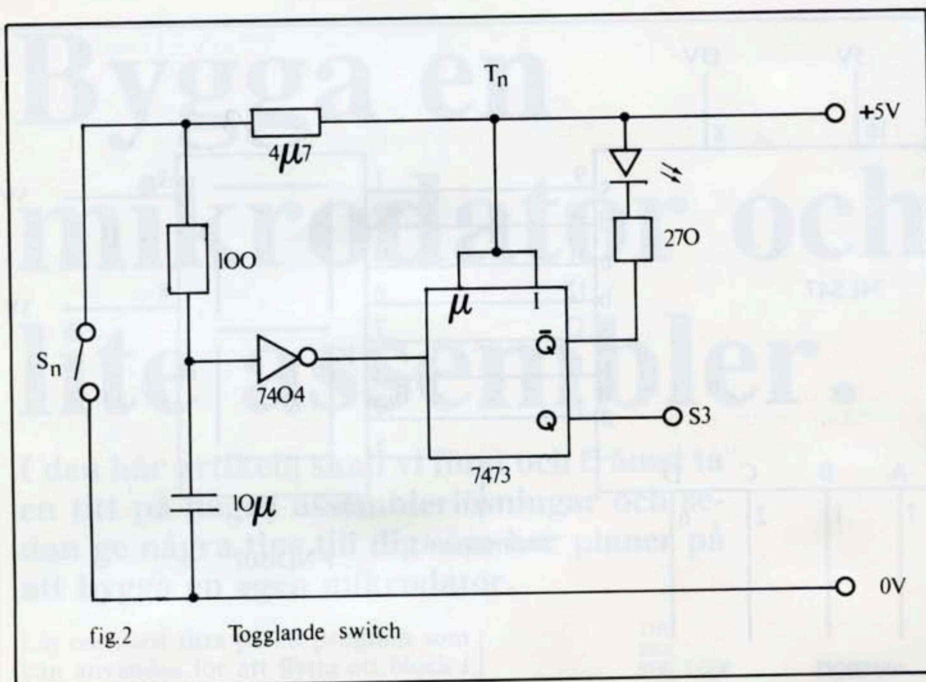


fig.2 Togglande switch

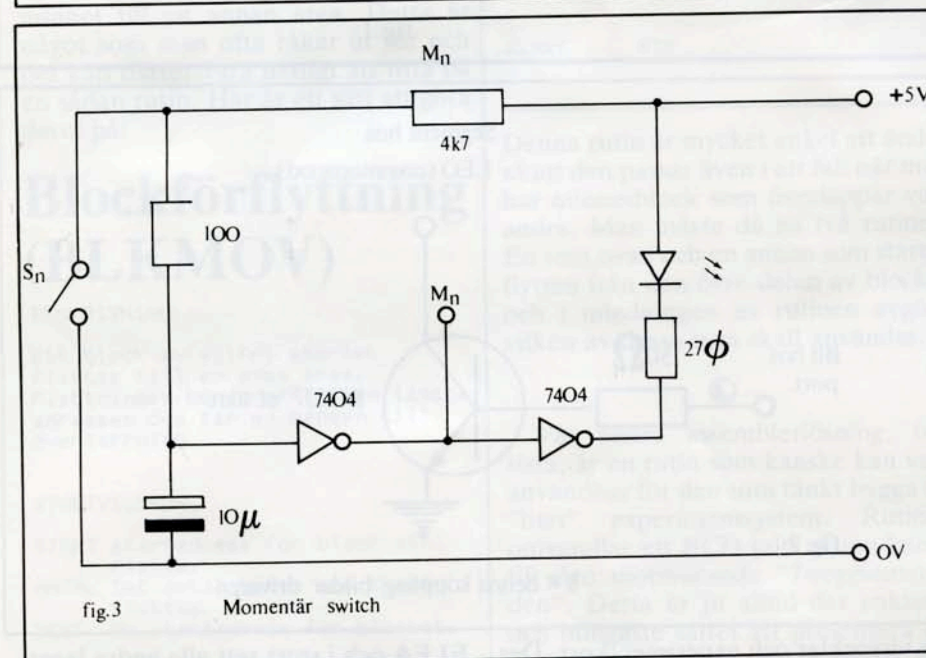


fig.3 Momentär switch

Nästa investering man bör göra är att inhandla ett oscilloskop och en bra voltmeter. Ett oscilloskop på ca 15 MHz är perfekt. Tyvärr så är ju oscilloskopen så pass kostsamma att de flesta inte har råd med ett. Man kan naturligtvis klara sig utan men bör då i alla fall investera i en logikprob, tex bygget i Elektronik världen nr 2-84. Man måste helt enkelt ha något som indikerar vilka nivåer man har på vissa pinnar samt om klockan verkligen ger en puls ut och att denna går fram överallt.

I fig 2 finns en "togglande switch" som är mycket nyttig i många fall. Togglande betyder helt enkelt att den byter nivå på utgången när man trycker på knappen. Denna krets är till stor hjälp vid allt digitalt arbete och är inte så svår att "slänga" ihop heller.

I fig 3 ser Du en momentär switch. Dvs en krets som ger en "etta" ut när man sluter brytaren utan att ge en massa studsar (kontaktstuds ställer garanterat till med problem) som en enkel brytare alltid åstadkommer.

En signalgenerator med en TTL-signal ut är också till stor hjälp och den som beskrivits som bygge i ALLT OM ELEKTRONIK går utmärkt att använda.

Till alla mikroprocessorer behövs någon form av klocka, till 65:an skall detta vara en tvåfasig tingest på antingen 1MHz eller 2 MHz beroende på vilken modell av kretsen man har tänkt jobba med. En mycket bra koppling från applikationsmanualen till 6502:an hittar Du i fig 4.

Om man tycker det skulle vara bra med två akumulatorer och ett 16-bits indexregister så kan Du som kan assembler på 6502:an med lätthet jobba med 6800. Till denna kan Du direkt koppla en kristall (se fig 5). Instruktionsreportoaren för de två är nästan

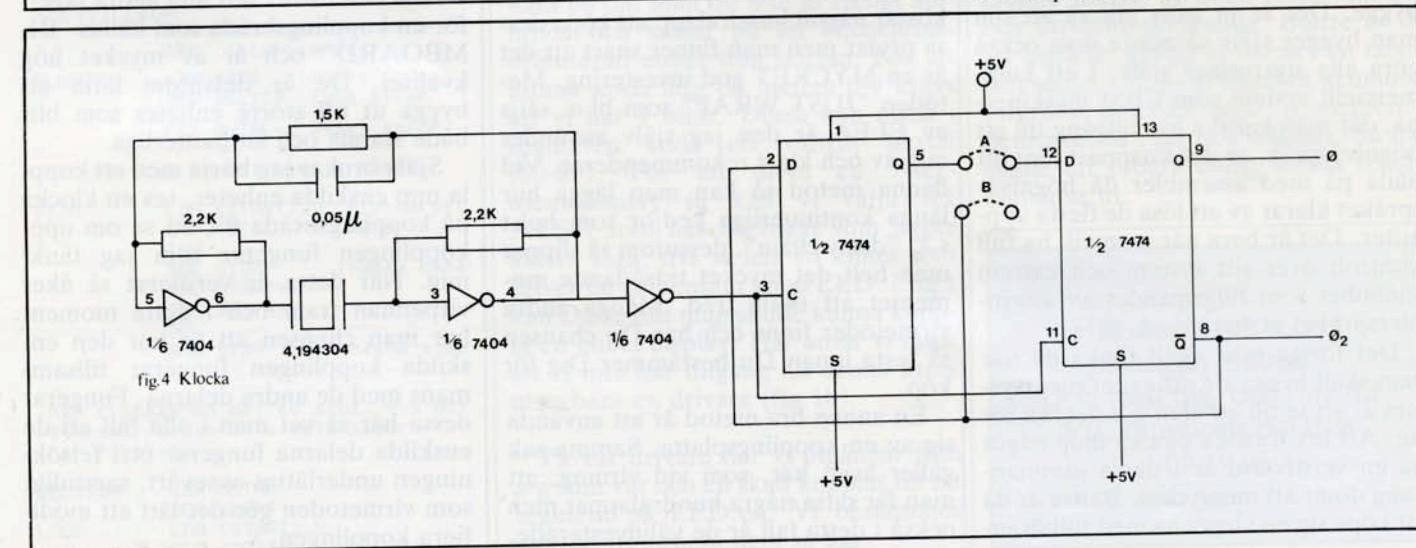


Fig.4 Klocka

lika och någon timmas studier av instruktionsreportoaren på 6800 gör att man behärskar även denna processor. Om Du lägger ned lite extra tid så får Du inte heller särskilt stora problem med en övergång från 6502:an till den formidabla och verkligen kraftfulla processorn 6809.

För alla nämnda processorer gäller att man måste tänka på att man måste "hissa" upp ett par av dess ben till +5V, detta sker lämpligen med hjälp av en resistans på ungefär 2.2K. Hos 6502:an är dessa ben IRQ,NMI,RDY,SO (kan lämnas öppen), SYNC och RESET. För reset kan dock en krets enligt fig 6 med fördel användas. Denna krets gör en RESET när strömmen slås på till datorn, vilket för att man på så sätt alltid får en väldefinierad programstart vid uppstarten av systemet. Vilket, om man tänker efter, är att föredra. Kretsen i fig 6 har också en brytare som gör det möjligt att göra en reset av datorn när man vill återstarta systemet.

Skall man mata många kretsar från sin mikrodator så skall man tänka på att adressbussen och databussen bara orkar driva en normal TTL last. Dvs skall man driva större laster så behövs någon form av buffertar. 74LS245, 74LS244, 8T28 och 8T97 är några man kan välja dessa kretsar är dock inte så lätt att få tag på med den kris och hamstring som råder på elektronikmarknaden.

När man gjort sin koppling kan det vara lämplig att först kontrollera att man har spänning till alla kretsar innan man sätter i kapslarna. Nästa steg är att sätta i kapslarna för klockan och kolla att denna går fram överallt och har de rätta nivåerna.

Sätt sedan i mikroprocessorn och kontroller att alla nivåer på dess pinnar ser ut att ligga normalt (IRQ,RESET etc skall tex ligga höga). Sätt sist i resten av kretsarna och håll tummarna för att allt fungerar. Gör det inte det, vilket det oftast inte gör, är det dags att börja felsöka. Vid felsökningen kan det löna sig att kolla så att man inte har några ledningar som det är avbrott på. Har man inte det kan man återigen kolla de olika enheterna och gå mot lägre och lägre nivå tills man hittar felet. Något man absolut inte skall misstänka i första hand är att "det är en IC som är felaktig". Det är mycket sällan som detta är fallet och det är bara att erkänna att felen i de absolut flesta fallen beror på begränsningarna hos oss konstruktörer eller med ett annat ord SLARV.

Jag hoppas att denna artikel i alla fall har givit något och att dessa hårdvarutips kan rädda i alla fall en från ett par timmars missökande. Assembler är inget som man lär sig över en natt så misströsta inte om Du tycker att Du inte kan något ännu. Det är endast genom övning och åter övning som man kan lära sig tekniken fullständigt. Detsamma torde gälla hårdvarupulandet. Det finns många bra böcker på området och Du bör inte ha något större problem att hitta någon som faller Dig i smaken. En bra svensk bok i ämnet har jag dock ännu inte sett men en sådan lär väl kanske någon gång se sitt ljus. Vad det gäller mikrodatorkonstruktioner så bygg på. Det är inte farligt att TÄNKA själv och testa egna idéer och kopplingar. I och med att kretsarna blir billigare och billigare så behöver det heller inte bli särskilt dyrt. Så bygg på.

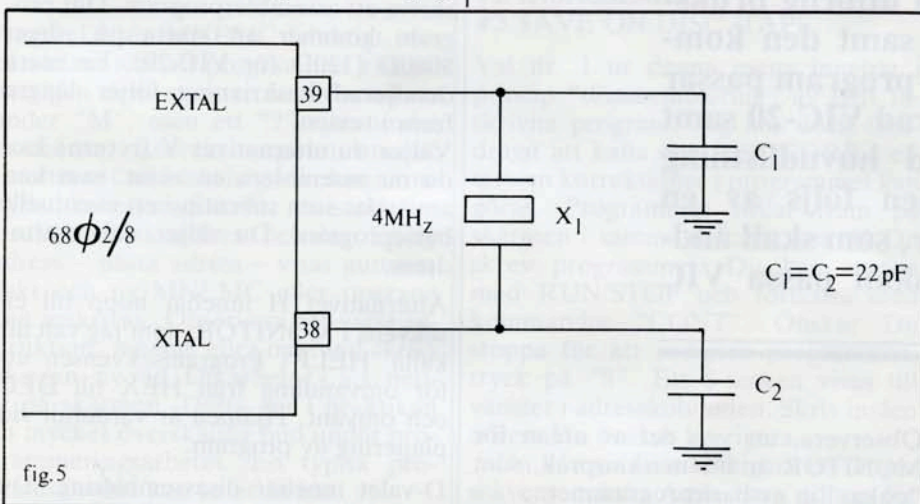


fig.5

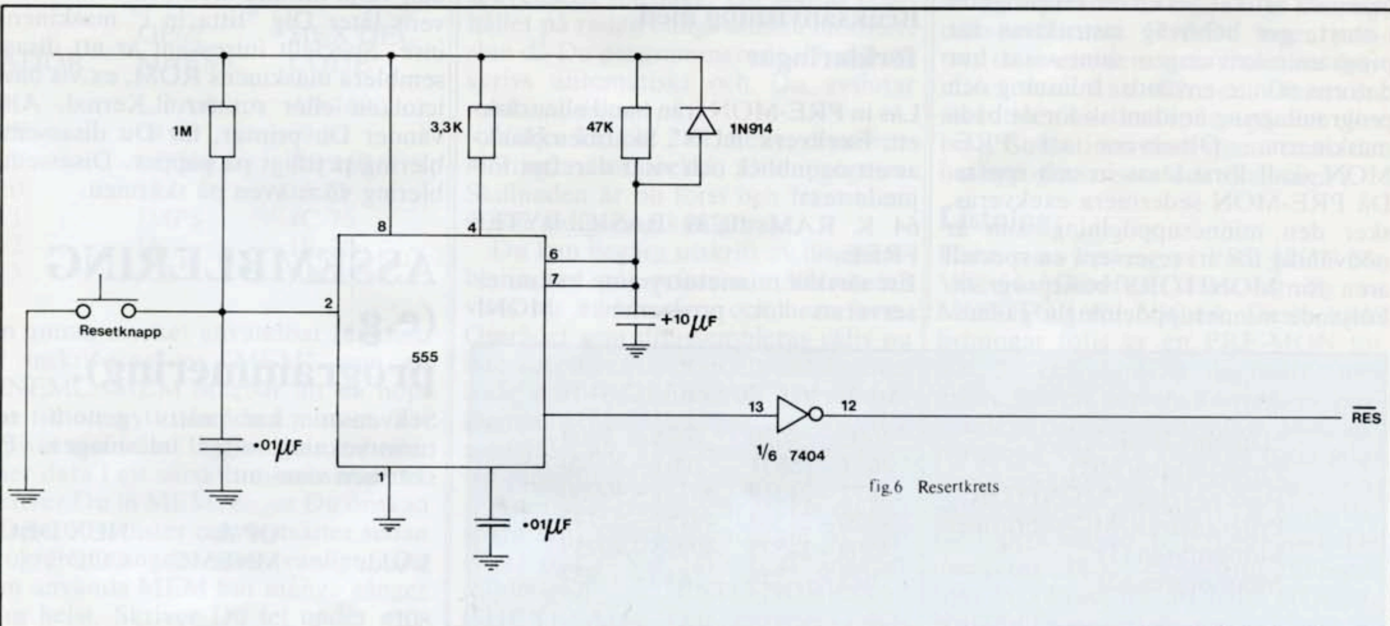


fig.6 Resertkrets

Maskinkods-laddare, del 2

I VIC-Rapports aprilnummer beskrevs en maskinkodsladdare för en expanderad VIC-20. Artikeln saknade erforderlig programlistning, ett måste för att programmet skall bli användbart. I avsikt att reparera missödet, presenteras nu en utförlig bruksanvisning till programmet samt den kompletta listningen. Beskrivna program passar både en med 16-K expanderad VIC-20 samt storebror VIC-64. Bifogad huvudlistning visar 64:ans program, men följs av en listning av de programrader, som skall ändras, för att programmet skall passa VIC 20-maskinen.

För att få full insikt i programmets struktur och möjligheter, måste aprilnumrets artikel läsas först. Nämnda artikel beskriver ett program i stort, ger behövlig instruktion för programinskrivningen samt visar hur datorns minne används. Inläsning och programlagring är identisk för de båda maskinerna. Observera att PRE-MON skall först läsas in och sparas. Då PRE-MON sedermera exekveras, sker den minnesuppdelning som är nödvändig för att reservera en speciell area för MONITORS basicprogram. Följande minnesuppdelningar gäller:

Observera, att viss del av arean för MONITOR är för maskinspråk, som "pokas" in av basicprogrammet.

Bruksanvisning med förklaringar

Läs in PRE-MON från band eller diskett. Exekvera 'RUN'. Skärmen blankas ett ögonblick och visar därefter följande text:
64 K RAM 10239 BASIC BYTES FREE.
Ett särskilt minnesutrymme har nu reserverats för programmet MONITOR. Då du nu läser in MONITOR, läses programmet till just detta område. Starta programmet med 'RUN' och på skärmen anges "WAIT FOR LOADING". Maskinspråksrutinerna i MONITOR 'bokas' nu in. Sekvensen avslutas med följande skärmbild:

```
[MONITOR]
ASSEMBLE = [RETURN]
BASICLOAD[Y]/HELP[H]/DISASSEMBLE[D]?
```

Genom att exekvera return, Y, H eller D kan du utföra följande:
Anger du endast [return] kan Du nu skriva ett assemblerprogram. Ditt program kommer att starta på adress $080D (120D för VIC-20). En mera detaljerad beskrivning följer längre fram i texten.
Väljer du alternativet Y (return) kan du nu assemblera en rutin, som kan användas som subrutin i ett eventuellt basicprogram. Du väljer själv startadress.

Alternativet H innebär hopp till en sekvens i MONITOR, som jag valt att kalla 'HELP'. Programsekvensen utför omvandling från HEX till DEC och omvänt. Hjälpen är värdefull vid planering av program.

D-valet innebär disassemblering. Du kan välja önskad area och denna sekvens låter Dig "titta in i" maskinens inre. Speciellt intressant är att disassemblera maskinens ROM, ex.vis basictolken eller rutiner i Kernal. Använder Du printer, får Du disassemblering prydligt på papper. Disassemblering visas även på skärmen.

	Adress VIC-20	d:o VIC-64
"Dummy Basic"	4608 – 4620	2048 – 2060
Programarea(MC)	4621 – 15871	2061 – 30720
Basicprogram, MONITOR	15872 – 24319	30721 – 40950
Programarea, MC-subrutiner	24320 – 24568	49152 – 53247
Monitorarea (1)	24569 – 24575	40953 – 40959
Monitorarea (2)	673 – 756	840 – 927

ASSEMBLERING (e.g. programmering).

Sekvensen har nåtts genom returntryckning efter inläsningen. På skärmen visas nu:

$Addr	OP/& MNEMC	HEX DEC CODE
80D		

Markören är placerad under "M" i MNEMC. Skriv in en mnemc och exekvera med return. Observera att programmet kräver mnemcs i format fyrställig kod (4 tecken). Den fjärde symbolen i mnemc-gruppen anger adresseringsmode. De använda tecknen är "A, #, $, X, Y, +, ., I. En komplett förteckning ser ut som följer:

Adresseringsmode:	Skrivs med:	
Relative & Impled	.	(ex: TAX.)
Accumulator	A	(ex: ASLA)
Immediate	#	(ex: LDA#)
Zero Page	Z	(ex: STAZ)
Absolute	$	(ex: LDA$)
Absolute, X	X	(ex: STAX)
Absolute, Y	Y	(ex: STAY)
Indirect, X	1	(ex: LDA1)
Y, Indirect	2	(ex: STA2)
Jump indirect	I	(ex: JMPI)

Anger Du en MNEMC felaktigt, dvs på sådant sätt att programmet ej känner igen den, stannar markören kvar under "M", men ett "?" syns under markören. Korrekt angiven mnemc resulterar i att båda högra kolumnerna ifylls i reverserad skrift, utvisande instruktionens kod, hex. och dec. En ny adress – nästa adress – visas automatiskt och ny MNEMC eller operand kan inskrivas. Observera att varje instruktion, mnemc eller operand skrivs på egen, ny rad. Du arbetar s.a.s. nedåt på skärmen. Detta ger i praktiken en mycket överskådlig bild under programmeringsarbetet. En typisk programsekvens kan se ut så här:

$ADDR.	OP/& MNEMC	HEX DEC CODE
80D	LDA#	A9 169
80E	09	9 9
80F	LDY#	A0 160
810	15	15 21
811	JMP$	4C 76
812	1E	1E 30
813	AB	AB 171

En annan mycket användbar funktion är inskrivning av "MEM" som en MNEMC. MEM innebär att du hoppar till ett nytt område i minnet. Ett typiskt sätt är att lägga texter, tabeller eller data i ett särskilt minnesområde. Skriver Du in MEM, anger Du önskad adress till vänster och fortsätter sedan programmeringen som vanligt. Du kan använda MEM hur många gånger som helst. Skriver Du fel under programmeringen och upptäcker detta några rader upp på skärmen, skriv in en £ (snabel-a) och raden ovanför markören suddas bort. Du kan på detta sätt stega bakåt och radera i programmet så långt Du önskar. En annan sak att beakta är att operander skrivs in i HEX-form. Samma gäller numeriska värden (siffror) som används i matematiska operationer. Siffror och bokstäver, som ingår i bl.a. textsträngar skrivs i vanlig form, dvs A B C .... 1 2 3 etc. För att underlätta vid just skrivandet av text finns på pil till vänster koden för CR (CHR (13) och på pil uppåt finns space (mellanslag=CHR$(20)). Du avslutar ditt arbete i denna sekvens (assembleringen) med att skriva in en asterisk (*). En meny med utseendet enligt nedan presenteras då.

PROGRAM ENDED!

#1 LIST/CORR.
#2 DISASSEMBLE
#3 SAVE ON DISC./TAPE

Val nr. 1 ur denna meny innebär i princip "disassemblering" av Ditt inskrivna program. Jag har dock föredragit att kalla den LIST/CORR., eftersom korrektioner i programmet kan göras. Programmet rullar fram på skärmen i samma format, som då Du skrev programmet. Du kan stoppa med RUN/STOP och fortsätta med kommandot "CONT". Önskar Du stoppa för att redigera programmet, tryck på "S". Ett $-tecken visas till vänster i adresskolumnen. Skriv in den adress Du önskar korrigera.

Du kan givetvis korrigera en längre sekvens också. Ange bara adress för sekvensens startlägg. Du ändrar innehållet på raden enligt samma mönster, som då Du programmerade. Ny adress skrivs automatiskt och Du avslutar med asterisken.

Väljer Du nr. 2 ur meny, disassemblering, visas också Ditt program. Skillnaden är nu först och främst, att ingen redigering kan göras.

Du kan begära utskrift av disassembleringen på printern. Adresserna visas nu i både dec. och hex. form. Området som disassembleras väljs nu inte automatiskt, utan Du måste ange både start- och slutadress i hex-form. Denna programsekvens är identisk med valet av "D" i menyn, som visas vid programstarten.

Alternativ tre innebär att Du nu kan spara Ditt assemblerprogram på tape eller disk. Val av 3:an innebär följdfrågan ENDADDRESS+1 ($HEX). Ange Ditt programs slutadress +1. Skärmen blir blank. Skriv det vanliga save-kommandot med angivet programnamn och enhetsnummer 8, om det gäller diskdriven. Då programmet lagrats, skrivs READY ut på skärmen. Vill Du nu återgå till att använda Ditt hjälpprogram MONITOR igen, skriv SYS908 för VIC 64 och SYS697 för VIC-20. SYS-kommandot skall därefter följas av "RUN".

Ditt sparade program kan laddas med "LOAD" och tack vare den s.k. Dummy Basicen startas med "RUN". Du skall vidare lägga märke till, att det endast är "rena maskinprogram" som kan sparas på det här sättet. Maskinprogram, som skall ingå som subrutiner i basicprogram, bokas in av basic. Det exakta användningssättet visas nu.

Skriva rutiner till Basicprogram:

Ur huvudmenyn väljer Du Y (return). På skärmen visas lämpligt minnesutrymme för Dina subrutiner. Ange avsedd startadress. Skärmen visar nu samma utseende och format, som vid vanlig assemblering. Den startadress Du valt, visas längst till vänster. Det är från denna adress, som uppräkningen nu sker. Programmet avslutas med asterisken och undermenyn visas. Denna meny har nu fått ett nytt alternativ, nr 4, som säger "BASIC LOAD,PTR". Sekvensen innebär att utskrift på printern visar de värden, som skall pokas in av ett basicprogram. Du får angivet start och slutadress i decimal form. Avsikten är att Du skall kunna skriva en FOR...TO...sats enligt visad adress och läsa in maskinspråksrutinen som datasatser. Formatet på skrivaren är lagt enligt lämpligt data-satsformat. Har Du ingen printer, disassemblera och anteckna decimalvärdena från skärmen. Observera att rutinen inte kan sparas på disk eller tape. Subrutinen skall sparas inom det basicprogram, som den skall ingå i.

Listning:

Först visas listning av PRE-MON för VIC-64. Därefter visas listning av MONITOR för VIC-64. Dessa båda listningar följs av en PRE-MON för VIC-20 och därefter de rader, som måste ändras för att konvertera programmet till en med minst 16-K expanderad VIC-20. Du skall lägga märke till aprilnumrets listförklaringar vad gäller grafiska symboler. De underlättar inskrivningen av programmet. De markerar bl.a. cursorförflyttningar, som är viktiga för att hålla ett visst, bestämt format på skärmen. Lycka till med assembleringen. →

<PRE-MON FöR CBM-64>

```
0 REM:*********************************
1 REM:*****< PRE-MON FöR CBM-64 >*****
2 REM:*********************************
5 FORN=840TO853
10 READA:POKEN,A:NEXTN
15 SYS840
20 DATA169,120,141,130,2,169,158,133,52,133,56,32,248,252
25 REM:PRE-MON RESERVERAR BASICAREAN FöR SJÄLVA PROGRAMMET 'MONITOR'.

READY.
```

<MONITOR FöR CBM-64>

```
0 REM:*********************************
1 REM:*****< MONITOR FöR CBM-64 >*****
2 REM:*********************************
3 IFPEEK(884)=169THENPRINT"■▢":POKE53261,0:POKE53280,0:GOTO9
4 POKE53280,0:POKE53281,0:POKE52,159:POKE56,159:PRINT"■▢";"WAIT FOR LOADING !"
5 FORN=854TO883:READA:POKEN,A:NEXTN:GOTO8
6 DATA162,0,189,103,3,157,0,8,232,224,13,240,3,76,88,3,96
7 DATA0,13,8,0,0,158,50,48,54,49,0,0,0
8 GOSUB300:PRINT"▢"
9 PRINTTAB(4)"< *MONITOR* >":PRINTTAB(7)"────────":PRINT:PRINT"*ASSEMBLE*=<RETURN
>":PRINT:PRINT
10 INPUT"BASICLOAD(Y)/HELP(H)/DISASSEMBLE(D)";Z$:GOTO94
25 K=16:B$=""
30 Y=INT(X/K):Z=X-Y*K:X=Y
40 X$=RIGHT$(STR$(Z),1):IFZ>9THENX$=CHR$(Z+55)
50 B$=X$+B$:IFX=0THENRETURN
55 GOTO30
60 K=16:A=0:G=0:N=LEN(A$)-1
70 FORI=NTO0STEP-1:A=A+1
75 Z=ASC(MID$(A$,A,1))-48:IFZ>16THENZ=Z-7
80 IFZ>15THENPRINT"■■■■■■■■■■                    ■■■■■■■■■■■■■■■■";:GOTO103
90 Z=Z*K↑I:G=G+Z:NEXTI:RETURN
94 IFZ$="D"THEN600
95 IF(Z$<>"Y")AND(Z$<>"H")THEN100
96 IFZ$="H"THEN900
97 PRINT"▢BASIC-LOAD":PRINT"────────────":PRINT:PRINT"FREE AREAS=49152-53247":PRIN
T
99 INPUT"STARTADDRESS(DEC)";L:L=L-1:PRINT"▢":GOTO101
100 PRINT"▢":L=2060:PRINTTAB(6)"ASSEMBLER":PRINTTAB(6)"──────────":PRINT
101 PRINTTAB(7)"OP/&   ▢■HEX DEC":PRINT"■■ADDR. MNEMC  CODE":PRINT"────────────
────────":F=L+1
102 M$=" ":OPEN9,0:L=L+1:X=L:GOSUB25:PRINTB$;TAB(7);
103 INPUT#9,M$:IFM$="@"THENPRINT"■■ ▢■                    ■■■■■■■■■■■■■■■■■▢":L=L-2:CLOSE9
:GOTO102
104 IFM$="MEM"THENPRINT:PRINT"▢MEM                    ■■■■■■■■■■":INPUT#9,A$:GOSUB60:L=G-1:PRI
NT"▢":CLOSE9:GOTO102
```

→

```
105 IFM$="↑"THENM$="SP;;"
106 IFM$="←"THENM$="CR;;"
107 IFM$="*"THENU=L-1:CLOSE9:GOTO120
108 IFLEN(M$)=4THENGOSUB400:GOTO112
109 IFLEN(M$)=1THENX=ASC(M$):GOSUB25:K$=B$:GOTO112
110 IFLEN(M$)=2THENK$=M$:GOTO112
111 POKE40957,94:GOTO403
112 PRINT"▒◥"TAB(13)K$" ";:A$=K$:GOSUB60:PRINTTAB(16)G;"▀◥":POKEL,G:CLOSE9:GOTO1
02
120 PRINT"▚PROGRAM ENDED !":PRINT
125 PRINT"#1 LIST/CORR.":PRINT
130 PRINT"#2 DISASSEMBLE":PRINT
135 IFZ$="Y"THENPRINT"#4 BASIC-LOAD(PTR)":PRINT:GOTO150
140 PRINT"#3 SAVE ON DISC./TAPE":PRINT
150 VA=0:INPUT"VAL #";VA:ONVAGOTO500,600,700,800:RUN
300 FORV=884TO927:READA:POKEV,A:NEXTV
301 FORN=29184TO30268
302 READA:POKEN,A:NEXTN
303 FORI=30464TO30630
304 READA:POKEI,A:NEXTI
305 RETURN
306 DATA169,1,133,43,133,172,169,8,133,44,133,173
307 DATA141,130,2,169,0,133,45,169,0,133,46,0
308 DATA169,0,133,45,169,0,133,46,169,120,133,44,141,130,2,169,1,133,43,96
309 DATA67,82,59,59,48,68,1,00,00,00,00,00,0
310 DATA65,68,67,35,54,57,2,65,68,67,90,54,53,2,65,68,67,43,55,53,2
311 DATA65,68,67,36,54,68,3,65,68,67,88,55,68,3,65,68,67,89,55,57,3
312 DATA65,68,67,49,54,49,2,65,68,67,50,55,49,2,65,78,68,35,50,57,2,65,78,68,90,
50,53,2
313 DATA65,78,68,43,51,53,2,65,78,68,36,50,68,3,65,78,68,88,51,68,3,65,78,68,89,
51,57,3
314 DATA65,78,68,49,50,49,2,65,78,68,50,51,49,2,65,83,76,65,48,65,1,65,83,76,90,
48,54,2
315 DATA65,83,76,43,49,54,2,65,83,76,36,48,69,3,65,83,76,88,49,69,3,66,67,67,46,
57,48,2
316 DATA66,67,83,46,66,48,2,66,69,81,46,70,48,2,66,73,84,90,50,52,2,66,73,84,36,
50,67,3
317 DATA66,77,73,46,51,48,2,66,78,69,46,68,48,2,66,80,76,46,49,48,2,66,82,75,46,
48,48,1
318 DATA66,86,67,46,53,48,2,66,86,83,46,55,48,2,67,76,67,46,49,56,1,67,76,68,46,
68,56,1
319 DATA67,76,73,46,53,56,1,67,76,86,46,66,56,1,67,77,80,35,67,57,2,67,77,80,90,
67,53,2
320 DATA67,77,80,43,68,53,2,67,77,80,36,67,68,3,67,77,80,88,68,68,3,67,77,80,89,
68,57,3
321 DATA67,77,80,49,67,49,2,67,77,80,50,68,49,2,67,80,88,35,69,48,2,67,80,88,90,
69,52,2
322 DATA67,80,88,36,69,67,3,67,80,89,35,67,48,2,67,80,89,90,67,52,2,67,80,89,36,
67,67,3
323 DATA68,69,67,90,67,54,2,68,69,67,43,68,54,2,68,69,67,36,67,69,2,68,69,67,88,
68,69,3
324 DATA68,69,88,46,67,65,1,68,69,89,46,56,56,1,69,79,82,35,52,57,2,69,79,82,90,
52,53,2
325 DATA69,79,82,43,53,53,2,69,79,82,36,52,68,3,69,79,82,88,53,68,3,69,79,82,89,
53,57,3
```

→

```
326 DATA69,79,82,49,52,49,2,69,79,82,50,53,49,2,73,78,67,90,69,54,2,73,78,67,43,
70,54,2
327 DATA73,78,67,36,69,69,3,73,78,67,88,70,69,3,73,78,89,46,67,56,1,74,77,80,36,
52,67,3
328 DATA74,77,80,73,54,67,3,74,83,82,36,50,48,3,76,68,65,35,65,57,2,76,68,65,90,
65,53,2
329 DATA76,68,65,43,66,53,2,76,68,65,36,65,68,3,76,68,65,88,66,66,3,76,68,65,89,
66,57,3
330 DATA76,68,65,49,65,49,2,76,68,65,50,66,49,2,76,68,88,35,65,50,2,76,68,88,90,
65,54,2
331 DATA76,68,88,43,66,54,2,76,68,88,36,65,69,3,76,68,88,89,66,69,3,76,68,89,35,
65,48,2
332 DATA76,68,89,90,65,52,2,76,68,89,43,66,52,2,76,68,89,36,65,67,3,76,68,89,88,
66,67,3
333 DATA76,83,82,65,52,65,1,76,83,82,90,52,54,2,76,83,82,43,53,54,2,76,83,82,36,
52,69,3
334 DATA76,83,82,88,53,69,3,78,79,80,46,69,65,1,79,82,65,35,48,57,2,79,82,65,90,
48,53,2
335 DATA79,82,65,43,49,53,2,79,82,65,36,48,68,3,79,82,65,88,49,68,3,79,82,65,89,
49,57,3
336 DATA79,82,65,49,48,49,2,79,82,65,50,49,49,2,80,72,65,46,52,56,1,80,72,80,46,
48,56,1
337 DATA80,76,65,46,54,56,1,80,76,80,46,50,56,1,82,79,76,65,50,65,1,82,79,76,90,
50,54,2
338 DATA82,79,76,43,51,54,2,82,79,76,36,50,69,3,82,79,76,88,51,69,3,82,79,82,65,
54,65,1
339 DATA82,79,82,90,54,54,2,82,79,82,43,55,54,2,82,79,82,36,54,69,3,82,79,82,88,
55,69,3
340 DATA82,84,73,46,52,48,1,82,84,83,46,54,48,1,83,66,67,35,69,57,2,83,66,67,90,
69,53,2
341 DATA83,66,67,43,70,53,2,83,66,67,36,69,68,3,83,66,67,88,70,68,3,83,66,67,89,
70,57,3
342 DATA83,66,67,49,69,49,2,83,66,67,50,70,49,2,83,69,67,46,51,56,1,83,69,68,46,
70,56,1
343 DATA83,69,73,46,55,56,1,83,84,65,90,56,53,2,83,84,65,43,57,53,2,83,84,65,36,
56,68,3
344 DATA83,84,65,88,57,68,3,83,84,65,89,57,57,3,83,84,65,49,56,49,2,83,84,65,50,
57,49,2
345 DATA83,84,88,90,56,54,2,83,84,88,43,57,54,2,83,84,88,36,56,69,3,83,84,89,90,
56,52,2
346 DATA83,84,89,43,57,52,2,83,84,89,36,56,67,3,84,65,88,46,65,65,1,84,65,89,46,
65,56,1
347 DATA84,89,65,46,57,56,1,84,83,88,46,66,65,1,84,88,65,46,56,65,1,84,88,83,46,
57,65,1
348 DATA83,80,59,59,50,48,1,69,83,67,59,49,66,1,73,78,88,46,69,56,1
350 DATA169,0,133,90,169,113,133
351 DATA91,160,255,162,0,189,249
352 DATA159,32,62,119,209,90,240
353 DATA15,165,91,201,119,208,6
354 DATA169,94,141,253,159,96,76
355 DATA10,119,232,224,4,240,3
356 DATA76,12,119,32,62,119,177
357 DATA90,157,249,159,232,224,7
358 DATA240,3,76,45,119,96,192
359 DATA255,208,5,160,0,230,91
```

```
360 DATA96,200,96,234,234,234,234
361 DATA234,234,234,169,0,133,90
362 DATA169,113,133,91,160,255,162
363 DATA0,189,249,159,32,62,119
364 DATA209,90,240,3,76,90,119
365 DATA232,224,2,240,3,76,92
366 DATA119,32,62,119,169,4,209,90
367 DATA16,6,32,156,119,76,90
368 DATA119,177,90,157,249,159,32
369 DATA156,119,32,156,119,32,156
370 DATA119,177,90,232,157,249,159
371 DATA224,6,240,3,76,139,119
372 DATA96,192,0,208,5,160,255
373 DATA198,91,96,136,96
400 POKE40953,ASC(M$):POKE40954,ASC(MID$(M$,2,1))
401 POKE40955,ASC(MID$(M$,3,1)):POKE40956,ASC(RIGHT$(M$,1))
402 SYS30464:K$=CHR$(PEEK(40957))+CHR$(PEEK(40958))
403 IFPEEK(40957)=94THENL=L-1:PRINT"▮▮▮▮?    ▯":CLOSE9:GOTO102
404 RETURN
405 M=0:POKE40953,ASC(C$):POKE40954,ASC(RIGHT$(C$,1))
406 SYS30544:D$=CHR$(PEEK(40959))+CHR$(PEEK(40958))+CHR$(PEEK(40957))+CHR$(PEEK(
40956))
408 M=PEEK(40955)
409 RETURN
500 PRINT"▯":PRINTTAB(4)"LISTER/CORR.":PRINTTAB(4)"─────────────":PRINT:X$=""
501 IF(ZZ=1)AND(U=>P)THEN508
502 IFZZ=1THENU=P:GOTO609
508 P=U:GOTO609
550 OPEN9,0:PRINT"$▮";:INPUT#9,A$:GOSUB60:L=G-1:M=0:ZZ=1:PRINT"▮▮▮▮▮                    ▮▮▮▮
▮"
551 CLOSE4:CLOSE9:GOTO102
600 PRINT"▯";"DISASSEMBLE":PRINT
601 INPUT"HAR DU PRINTER <J/N>";Y$:IFY$="N"THENY$="":PJ=1:PRINT"▯":GOTO605
602 OPEN4,4:PRINT
605 INPUT"STARTADDRESS.(HEX $)";A$:GOSUB60:F=G:PRINT
606 INPUT"END ADDRESS .(HEX $)";A$:GOSUB60:U=G:PRINT"▯▯DISASSEMBLER":PRINT"──────
─────"
607 IFVA<>1THENPRINT:PRINT" DEC.ADDR.";TAB(10)"HEX.ADDR.";TAB(20)"MNEMC";TAB(30)
"CODE:"
608 PRINT"──────────────────────────────────────"
609 D$="":M=0
610 FORE=FTOU:B=PEEK(E):X=E:GOSUB25:A$=B$:X=B:GOSUB25:C$=B$:IFLEN(C$)<2THENC$="0
"+C$
679 IFM>0THEN682
680 GOSUB405
682 M=M-1:IFVA=1THEN685
684 PRINTE;TAB(10)A$;TAB(20)D$;TAB(30)"▮"C$"   "B;"▮":D$="":GOTO689
685 PRINTA$;TAB(7)D$;"▮"TAB(13)C$" ";TAB(16)B;"▮":GETX$:IFX$="S"THEN550
686 IFVA=1THEND$="":NEXTE:INPUTY$:GOTO120
689 IFPJ=1THEN695
690 PRINT#4,E;CHR$(16)"11"A$;CHR$(16)"21"D$;CHR$(16)"31"C$;CHR$(16)"38"B:D$=""
695 NEXTE:Y$="":PJ=0:CLOSE4:INPUT"CONT.(Y)";Y$:IFY$="Y"THEN600
696 GOTO120
700 PRINT"▯";"SAVE ON TAPE/DISC!":PRINT
710 INPUT"END ADDR.(HEX$)+1";U$:SYS854
```

→

```
720 POKE909,PEEK(45):POKE913,PEEK(46)
730 A$=RIGHT$(U$,2):GOSUB60:POKE900,G:A$=LEFT$(U$,2):GOSUB60:POKE904,G:SYS884
745 REM:*************************************
750 REM:ÅTERSTART EFTER 'SAVING'= SYS908 !!!
755 REM:*************************************
800 PRINT"JBASIC-LOADER":PRINT
805 OPEN4,4:A=0:PRINT#4,"BASIC-LOADER":PRINT#4,"————————————";CHR$(13)
806 PRINT#4,F"-"U
810 FORN=FTOU:PRINT#4,PEEK(N);
815 A=A+1:IFA=15THENA=0:PRINT#4,CHR$(13)
820 NEXTN:PRINT#4:CLOSE4:INPUTZ$:X$="":GOTO120
900 PRINT"JHELP-PROGRAM":PRINT"————————————":PRINT
910 PRINT"#1 DEC ->  HEX":PRINT
915 PRINT"#2 HEX ->  DEC":PRINT
920 W=0:INPUT"VAL#";W
921 ONWGOTO930,940
925 RUN
930 PRINT"J";:INPUT"DEC:";X:GOSUB25:PRINT"JTaTaTaTaTaTaTaTaTaTaTaT";"=HEX:$"B$:PRINT:INPUTZZ
$:GOTO950
940 PRINT"J";:INPUT"HEX$:";A$:GOSUB60:PRINT"JTaTaTaTaTaTaTaTaTaTaTaTa=DEC"G:PRINT:INPUTZZ$
950 IFZZ$="RUN"THENRUN
955 GOTO921

READY.
```

<PRE-MON FöR VIC-20>

```
0 REM:*********************************
1 REM:*****< PRE-MON FöR VIC-20 >*****
2 REM:*********************************
5 FORN=711TO724
10 READA:POKEN,A:NEXTN
15 SYS711
20 DATA169,68,141,130,2,169,95,133,52,133,56,32,56,253
25 REM:PRE-MON RESERVERAR BASICAREAN FöR SJÄLVA PROGRAMMET 'MONITOR'.

READY.
```

<KONVERTERING TILL VIC-20(MONITOR)>

```
0 REM:*********************************
1 REM:*****< MONITOR FöR VIC-20 >*****
2 REM:*********************************
3 IFPEEK(673)=169THENPRINT"J":POKE36879,25:GOTO9
4 POKE36879,25:POKE52,95:POKE56,95:PRINT"J";"WAIT FOR LOADING !"
5 FORN=725TO754:READA:POKEN,A:NEXTN:GOTO8
6 DATA162,0,189,230,2,157,0,18,232,224,13,240,3,76,215,2,96
7 DATA0,13,18,0,0,158,52,54,50,49,0,0,0
8 GOSUB300:PRINT"J"
100 PRINT"J":L=4620:PRINTTAB(6)"ASSEMBLER":PRINTTAB(6)"——————————":PRINT
111 POKE24573,94:GOTO403
```

```
300 FORV=673TO709:READA:POKEV,A:NEXTV
301 FORN=15872TO16956
302 READA:POKEN,A:NEXTN
303 FORI=17152TO17318
304 READA:POKEI,A:NEXTI
305 GOTO375
306 DATA169,1,133,43,133,172,169,18,133,44,133,173
307 DATA141,130,2,169,0,133,45,169,0,133,46,0
308 DATA169,0,133,45,169,0,133,46,169,68,76,243,2
350 DATA169,0,133,0,169,61,133
351 DATA1,160,255,162,0,189,249
352 DATA95,32,62,67,209,0,240
353 DATA15,165,1,201,67,208,6
354 DATA169,94,141,253,95,96,76
355 DATA10,67,232,224,4,240,3
356 DATA76,12,67,32,62,67,177
357 DATA0,157,249,95,232,224,7
358 DATA240,3,76,45,67,96,192
359 DATA255,208,5,160,0,230,1
360 DATA96,200,96,234,234,234,234
361 DATA234,234,234,169,0,133,0
362 DATA169,61,133,1,160,255,162
363 DATA0,189,249,95,32,62,67
364 DATA209,0,240,3,76,90,67
365 DATA232,224,2,240,3,76,92
366 DATA67,32,62,67,169,4,209,0
367 DATA16,6,32,156,67,76,90
368 DATA67,177,0,157,249,95,32
369 DATA156,67,32,156,67,32,156
370 DATA67,177,0,232,157,249,95
371 DATA224,6,240,3,76,139,67
372 DATA96,192,0,208,5,160,255
373 DATA198,1,96,136,96
375 FORT=755TO765:READA:POKET,A:NEXTT
376 RETURN
377 DATA133,44,141,130,2,169,1,133,43,169,96
400 POKE24569,ASC(M$):POKE24570,ASC(MID$(M$,2,1))
401 POKE24571,ASC(MID$(M$,3,1)):POKE24572,ASC(RIGHT$(M$,1))
402 SYS17152:K$=CHR$(PEEK(24573))+CHR$(PEEK(24574))
403 IFPEEK(24573)=94THENL=L-1:PRINT"▮▮▮▮?    J":CLOSE9:GOTO102
404 RETURN
405 POKE24569,ASC(C$):POKE24570,ASC(RIGHT$(C$,1))
406 SYS17232:D$=CHR$(PEEK(24575))+CHR$(PEEK(24574))+CHR$(PEEK(24573))
407 D$=D$+CHR$(PEEK(24572))
408 M=PEEK(24571)
409 RETURN
410 REM:ÄNDRA ALLA 'TABBAR' I AVSNITTET 600-699
700 PRINT"J";"SAVE ON DISK./TAPE"
710 INPUT"ENDADDR.(HEX$+1)";U$:SYS725
720 POKE698,PEEK(45):POKE702,PEEK(46)
730 A$=RIGHT$(U$,2):GOSUB60:POKE689,G:A$=LEFT$(U$,2):GOSUB60:POKE693,G:SYS673
745 REM:******************************
750 REM:ÅTERSTARTA MONITOR MED SYS697
755 REM:******************************

READY.
```

Nybörjarskolan Del 6

Då var det dags att ta tjuren vid hornen igen och kasta sig in i BASIC's ljuvliga värld. Som vanligt kan vi inte svara för de följder som dessa sidor kan för den mer erfarne programmeraren och uppmanar alltså dig som tycker att du kan det här med BASIC att hoppa över de följande sidorna. I det förra numret så tittade vi på de möjligheter den sk IF...THEN strukturen gav oss. Med hjälp av denna så såg vi att vi kunde göra en test och fatta olika beslut beroende på utfallet av testen.

Subrutiner

Subrutiner kan man se som PROGRAM I PROGRAMMET. En subrutin kan t ex vara ett antal BASIC rader som väntar på att användaren (den som sitter och knapppar och knäpper) skall trycka på en knapp och sedan returnerar den tecken (den knapp) som ''Nisse vid terminalen'' tryckte ned. En subrutin som utför just detta kan se ut så här:

```
500 REM ******VÄNTA OCH TA IN TECKEN
510 PRINT"TRYCK PÅ EN TANGENT!"
520 GET A$ : IF A$="" THEN 520
530 RETURN
```

Denna subrutin tar in ett tecken från tangentbordet i A$. Dvs om man trycker ned ''L'' så kommer A$ bli lika med ''L''. Om däremot ingen tangent trycks ned så returneras en TOMSTRÄNG som vi redan tidigare har sett och i detta fall blir A$ = ''''.

IF...THEN satsen kontrollerar, som vi också tidigare har sett, om någon tangent har tryckts ned. Har den inte det så sker ett hopp tillbaka och ett nytt försök att ta in ett tecken görs.

Till slut kommer antagligen ''Nisse vid terminalen'' att trycka på en knapp och programmet fortsätter då till rad 530 där vi har RETURN. Detta är det ena av de två magiska orden när det gäller subrutiner.

Om du har slagit in programmet ovan så testa och kör det!

Om du ser får du ett felmeddelande sedan du tryckt på en tangent efter den inledande frågan.

Detta felmeddelande talar om att du kört en subrutin utan att ha ANROPAT den genom att använda det andra magiska ordet:

GOSUB

Både GOSUB och RETURN måste finnas med när man skriver subrutiner på precis samma sätt som ett IF kräver ordet THEN och FOR kräver TO och NEXT osv. När vi i subrutinen kommer till rad 530 så försöker programmet att göra ett hopp tillbaka till ordet efter det där anropet av subrutinen skedde. Låt oss titta på detta. Lägg till:

10 GOSUB 500

till ditt program dvs programmet får utseendet:

```
10  GOSUB 500
500 REM ******VÄNTA OCH TA IN TECKEN
510 PRINT"TRYCK PÅ EN TANGENT!"
520 GET A$   IF A$="" THEN 520
530 RETURN
```

Kör programmet åter igen!

Samma fel fortfarande eller hur?

Skillnaden den här gången var bara att man fick en uppmaning att mata in ett tecken två gånger.

Ser du vad vi gjort för fel?

Ja just det. Vi har glömt att skriva END efter subrutinen så att programmet stannar innan vi kommer ner hit. Vi lägger alltså till

20 END

och vårt lilla program får utseendet:

```
10  GOSUB 500
20  END
500 REM ******VÄNTA OCH TA IN TECKEN
510 PRINT"TRYCK PÅ EN TANGENT!"
520 GET A$ : IF A$="" THEN 520
530 RETURN
```

Kör det åter igen. Inga felmeddelanden den här gången eller hur?

Frågan är nu bara hur det hela fungerar? Ändrar vi rad 10 till:

10 GOTO 500

så vet vi sedan förut att GOTO hoppar till rad 500 och fortsätter att köra programmet bara med denna ändring så får vi återigen det, numera, ganska vanliga felet. TESTA! För att få det hela att fungera tillfredställande får vi ändra rad 530 också så att vi får ett hopp tillbaka. Vi tar alltså och ändrar rad 530 till:

530 GOTO 20

och programmet har fått utseendet:

```
10  GOTO  500
20  END
500 REM ******VÄNTA OCH TA IN TECKEN
510 PRINT"TRYCK PÅ EN TANGENT!"
520 GET A$ : IF A$="" THEN 520
530 GOTO  20
```

Kör du programmet igen fungerar det precis som vårt tidigare exempel. Vad skiljer nu de två åt?

Med GOSUB radnr så får vi tydligen ett hopp till det radnr vi ger precis som GOTO radnr. Om vi använder GOTO och hade en RETURN i uttrycket så fick vi dock inte programmet att hoppa tillbaks och vi fick istället ersätta RETURN med GOTO 20 för att få detta att ske.

GOSUB hoppar alltså till det radnummer där en subrutin förväntas ligga. Denna subrutin skiljer sig från andra delar av programmet så tillvida att det avslutas med BASIC ordet RETURN. Detta ger ett hopp tillbaka till det BASIC ord som finns efter den GOSUB som ''kallade'' (ANROPADE, HOPPADE TILL) subrutinen. I vårt exempel så fanns dock GOSUB'en på en egen rad och därför hoppade programmet till raden efter anropet. Om du ändrar programmet till:

```
10  GOSUB 500:PRINT A$
20  END
500 REM ******VÄNTA OCH TA IN TECKEN
510 PRINT"TRYCK PÅ EN TANGENT!"
520 GET A$ : IF A$="" THEN 520
530 RETURN
```

så bör även tvivlaren övertygas om att så är verkligen fallet. Kom alltså ihåg EN GOSUB SKALL ALLTID FÖLJAS AV EN RETURN.

Varför i hela friden ska vi krångla till allt på detta sätt. Programmet ovan kunde vi ha ju ha skrivit:

```
500 REM ******VÄNTA OCH TA IN TECKEN
510 PRINT"TRYCK PÅ EN TANGENT!"
520 GET A$ : IF A$="" THEN 520
530 END
```

Det är helt riktigt. Som vanligt så överförenklar vi naturligtvis programmen för att det ska vara lätt att förstå vad som händer. Du kan istället studera följande program:

```
10  PRINT"(SHIFT+CLR/HOME)":REM RENSA SKÄRMN OCH LÄGG CURSORN I TOPPEN
20  FOR I=1 TO 60
30  PRINT "LOOP RÄKNARE =";I
40  IF (I=15 OR I=30) OR I=45 THEN GOSUB 500
50  NEXT I
60  END
500 REM ******VÄNTA OCH TA IN TECKEN
510 PRINT"TRYCK PÅ EN TANGENT!"
520 GET A$ : IF A$="" THEN 520
530 PRINT"(SHIFT+CLR/HOME)":REM RENSA SKÄRMN OCH LÄGG CURSORN I TOPPEN
540 RETURN
```

Som du ser så kallar vi på subrutinen tre gånger i programmet för att riktigt ordentligt kunna följa det värde loopindex har. Om vi inte hade använt oss av en subrutin så skulle programmet få utseendet:

iii) Använd subrutiner flitigt. På detta sätt så delar man upp det programmeringsproblem man skall lösa i många små enheter som var för sig blir enkla att lösa.

```
10  PRINT"(SHIFT+CLR/HOME)":REM RENSA SKäRMN OCH LäGG CURSORN I TOPPEN
20  FOR I=1 TO 15
30  PRINT "LOOP RäKNARE =";I
40  NEXT I
50  PRINT"TRYCK På EN TANGENT!"
60  GET A$ : IF A$="" THEN 60
70  PRINT"(SHIFT+CLR/HOME)":REM RENSA SKäRMN OCH LäGG CURSORN I TOPPEN
80  FOR I=16 TO 30
90  PRINT "LOOP RäKNARE =";I
100 NEXT I
110 PRINT"TRYCK På EN TANGENT!"
120 GET A$ : IF A$="" THEN 120
130 PRINT"(SHIFT+CLR/HOME)":REM RENSA SKäRMN OCH LäGG CURSORN I TOPPEN
140 FOR I=31TO 45
150 PRINT "LOOP RäKNARE =";I
160 NEXT I
170 PRINT"TRYCK På EN TANGENT!"
180 GET A$ : IF A$="" THEN 180
190 PRINT"(SHIFT+CLR/HOME)":REM RENSA SKäRMN OCH LäGG CURSORN I TOPPEN
200 FOR I=46TO 60
210 PRINT "LOOP RäKNARE =";I
230 NEXT I
```

Nog måste man säga att det första programmet är lite snyggare. Man skall naturligtvis inte försumma den vinst i minnesåtgång som det förra ger heller. Subrutiner använder man sig alltså av när man har något i sitt programm som skall utföras flera gånger i programmet. Detta gäller naturligtvis inte om det som skall göras består av endast ett BASIC kommando eller om man kan göra det hela med hjälp av en loop.

Några tips som kan vara bra att hålla sig till när man använder sig av subrutiner i sina program är:

i) Inled alltid subrutinen med REM satser som talar om vad den aktuella rutinen utför. Detta är till ovärderlig hjälp när några månader har gått och man vill göra ändringar i sitt program. Har du inte råd med det extra minne som går åt så gör en anteckning om de ingående rutinerna på ett papper och spar. Du kommer att få igen detta flera gånger om.

ii) Ha endast en RETURN i varje subrutin. Dvs istället för att skriva:

```
1000  IF AX=1   THEN RETURN
1010  IF AX=10  THEN RETURN
1020  IF AX=100 THEN RETURN
1030  AX=AX*AX
1040  RETURN
```

så skriv:

```
1000  IF AX=1   THEN 1040
1010  IF AX=10  THEN 1040
1020  IF AX=100 THEN 1040
1030  AX=AX*AX
1040  RETURN
```

Förutom att programmet blir mer lättläst så förenklar detta eventuella senare ändringar.

Den del av programmet som anropar subrutiner med GOSUB brukar man ofta kalla HUVUDPROGRAM. Naturligtvis kan man också låta subrutinerna anropa andra subrutiner eller hoppa omkring på annat sätt. Man bör dock som vi påtalat tidigare begränsa sig i så stor utsträckning som möjligt från att hoppa omkring från den ena raden till den andra då programmet blir onödigt svårläst. Ett program som ser ut så här:

```
10  GOTO 120
20  PRINT "KALLE"
30  GOTO 90
40  PRINT "KAJSA"
50  GOTO 130
60  END
70  IF C=45 THEN 60
80  GOTO 100
90  GOSUB 100:GOTO 40
100 PRINT "JOBBIGT"
110 RETURN
120 GOTO 20
130 GOTO 60
```

gör ingen lycklig även om det fungerar. Naturligtvis är inte målet att skriva ett så snyggt program som möjligt, utan att lösa den programmeringsuppgift man har föresatt sig att lösa. Men det kan aldrig skada att tänka till åtminstone lite grann.

Två skumma kommandon

Ibland kan det hända att man vill stoppa sitt program på en bestämd rad för att sedan fortsätta igen. För detta ändamål så har man BASIC-ordet STOP som stoppar exekveringen av programmet på den rad den står och ger meddelandet BREAK IN radnr.

Vi kan testa detta genom följande program.

```
10 FOR I=1 TO 100
30 PRINT I
40 NEXT
50 STOP
60 FOR I=100 TO 1 STEP -1
70 PRINT I
80 NEXT
```

Kör du detta program så stannar det på rad 50 och
BREAK IN 50 skrivs ut.
Skriv nu
CONT
och tryck på RETURN. Programmet fortsätter eller hur?

Detsamma gäller om man trycker på tangenten märkt RUN/STOP kanske av misstag. Skriv CONT och tryck RETURN och programmet fortsätter som om inget har hänt. Du kan testa detta i programmet ovan genom att trycka på RUN/STOP direkt efter det att du startat programmet. Observera att det inte går att återstarta ett program som du editerat i dvs där du har ändrat om i programmet. Däremot så går det utmärkt att t ex printa variabler som du vill veta värdet hos eller utföra andra kommandon i direkt mod.

Några Tips

1) Om du vill ta bort en rad så skriv radnummer <RETURN>
och raden försvinner.
2) Om du vill flytta cursorn till början av nästa rad på skärmen utan att slå en RETURN dvs utan att tala om för datorn att det som står på raden ska behandlas.
Håll SHIFT nedtryckt och tryck på RETURN.
3) Om du vill flytta en rad till ett annat ställe så skriver du bara ett nytt radnummer och trycker sedan på RETURN. Du har fått en kopia på denna rad.
4) Om du har en BASIC rad som du är osäker på om du kanske behöver senare men vill ta bort för tillfället. Sätt en REM som första BASIC ord på raden. Detta REM kan du sedan ta bort om du skulle behöva raden igen.
5) Om alla rader på skärmen helt plötsligt blivit fördubblade. Gå då genast till sängs. Du har antagligen suttit ett par timmar för länge vid datorn.

I nästa nummer ska vi titta på de olika variabeltyper som vi har tillgång till hos VICKE. Som vanligt vill vi uppmana alla som har synpunkter på nybörjarskolan att höra av sig till redaktionen med ett brev eller vykort. Det kan vara allt från en regelrätt utskällning till önskemål om något du vill att vi ska ta upp.

CAR RACING VIC 20



Jyrki Hookanen tycker att VIC rapport är en intressant tidning och har skickat in ett spelförslag. Det är ett bilspel med både fart och spänning, och rekommenderas från 9 år. CAR-RACING kan köras med en oexpanderad VIC 20. Spelet går ut på att köra banan runt så många gånger som möjligt under en minut. Du har fyra växlar att utnyttja. Vid kollision övergår växeln automatiskt till neutral. Spelet är spärrat mot baklängeskörning genom mållinjen, vilket är ganska logiskt. Det kan vara ganska knepigt i början, men du kommer nog snart på tekniken. När du blir proffsig så kan du ändra banan till en högre svårighetsgrad.

```
1 REM***************
2 REM* BY J.HONKANEN*
3 REM* COPYRIGHT 84 *
4 REM***************
8 Z=0:X=0:Q=0:O=0
10 PRINT"♥"
13 TI$="000000"
15 POKE36879,9
17 E=36877
18 POKE 36878,5
20 W=36879
25 C=36876
30 D=300
49 PRINT"♥▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
50 PRINT"♥▒▒▒▒▒▒▒▒▒▒▒▒▒      ▒▒▒"
53 PRINT"♥▒▒▒▒       |       ▒"
55 PRINT"♥▒▒▒        |        ▒"
56 PRINT"♥▒▒         |        ▒"
57 PRINT"♥▒▒   ▒     |        ▒"
60 PRINT"♥▒          ▒▒     ▒  ▒"
62 PRINT"♥▒       ▒▒▒▒▒▒▒▒  ▒  ▒"
65 PRINT"♥▒     ▒▒▒▒▒▒▒▒▒▒     ▒"
70 PRINT"♥▒▒      ▒▒▒▒▒▒▒▒▒    ▒"
73 PRINT"♥▒▒▒      ▒▒▒▒▒▒▒▒    ▒"
75 PRINT"♥▒▒▒▒      ▒▒▒▒▒▒▒   ▒▒"
77 PRINT"♥▒▒▒▒▒▒     ▒▒▒▒▒▒▒  ▒▒"
79 PRINT"♥▒▒▒▒      ▒▒▒▒▒▒▒▒▒ ▒▒"
82 PRINT"♥▒▒▒     ▒▒▒▒▒▒▒▒▒▒   ▒"
85 PRINT"♥▒▒     ▒▒▒▒▒▒▒▒▒▒    ▒"
87 PRINT"♥▒▒    ▒▒▒▒▒▒▒▒▒▒▒    ▒"
90 PRINT"♥▒▒    ▒▒▒▒▒▒ . ▒▒    ▒"
92 PRINT"♥▒▒    ▒▒▒▒   .   ▒   ▒"
95 PRINT"♥▒▒     ▒▒    .       ▒"
97 PRINT"♥▒▒           .▒      ▒"
98—POKE8162,102:POKE8163,102
102 POKE8133,102:POKE8134,102:POKE8163,102 :POKE8155,102:POKE8156,102:POKE8154
02
103 POKE8157,102
106 FORI=1TO5
```

Programförklaring:

rad 8: nollställer olika variabler
rad 10-18: nollställer tiden och byter färgen till svart samt sätter på volymen.
rad 20-25: ger värde år ljudvariablar.
rad 49-133: ritar planen, märk dock att de sista raderna pokas för att undvika att skärmen börjar rulla.
rad 135: känner efter om växeln är neutral.
rad 150: testar om bilen har kolliderat med väggen.
rad 155: känner efter om man har passerat över mållinjen.
rad 152-153: skriver och känner efter om tiden är en minut.
rad 156: skriver ut antalet körda varv.
rad 157: kontrollerar om man har passerat halva varvet.
rad 150-220: huvudloop
rad 160: ritar ut bilen.
rad 170-200: kollar om någon tangent är nertryckt och ändrar värden.
rad 203-208: stänger av ljudet och väntar en stund beroende på vilken växel som är inne. Därefter sätts ljudet på igen.
rad 210: beräknar skärmpositionen för bilen.
rad 400-435: ändrar färger och sätter igång buller.
rad 436: stänger av ljudet.
rad 440-450: neutraliserar växlarna samt ritar ut bilen.
rad 500-512: skriver ut ditt slutresultat och högsta resultat samt väntar på att du skall trycka på "s".
rad 600: ökar varvantalet.
rad 700: passering av halva varvet.

Variablar:

E = ljudet (bruset)
Z = varv spärren
X = riktningspekare
Q = bilens position
O = antal varv
W = färgen
C = ljudet
D = växeln

Du styr åt vänster med :
Du styr åt höger med ;
Du styr upp med a (snabel-a)
Du styr ner med /
Om du har Å, Ä, Ö sats så skriv om följande rader.

```
180 IFA$="Ö"THEN X=-1
190 IFA$="Å"THEN X=-22
200 IFA$="Ä"THEN X=1

READY.
```

Jag hoppas att detta spel kan ge dig mycket nöje.

```
108 POKE8141+I,102
109 NEXTI
112 FORI=1TO22
124 POKE8163+I,102
126 NEXTI
127 POKE8141,102
130 FORI=1TO3
132 POKE8119+I,102
133 NEXTI
134 REM************************
135 IFZ=1THEN 440
150 IFPEEK(7757+Q)=102THEN GOTO400
152 PRINT"▩";TI$
153 IFTI$=>"000100"THENPOKEC,232:FORT=1TO700:NEXTT:POKEC,0:POKEE,0:GOTO500
155 IFPEEK(7757+Q)=93THENPOKE7757+Q,93:GOTO600
156 PRINT"TTTTTTTTTTTTT";D
157 IFPEEK(7757+Q)=46THENPOKE(7757+Q),46:GOTO700
160 POKE7757+Q,81
170 GETA$:IFA$="/"THEN X=22
171 IFA$="▮"THEND=140
173 IFA$="▤"THEND=50
175 IFA$="▥"THEND=25
179 IFA$="▦"THEND=8
180 IFA$=":"THEN X=-1
190 IFA$="@"THEN X=-22
200 IFA$=";"THEN X=1
203 POKEE,0
205 FORT=1TOD:NEXTT
207 POKE7757+Q,32
208 POKEE,232
210 Q=Q+X
220 GOTO150
399 REM************************
400 PRINT"▣"
402 POKEE,232
403 FORI=1TO3
405 POKEW,40
410 FORT=1TO50:NEXTT
415 POKEW,31
420 FORT=1TO50:NEXTT
425 POKEW,9
430 FORT=1TO50:NEXTT
435 NEXTI
436 Z=1
437 POKEE,0
438 GOTO15
440 X=0
445 D=300
450 POKE7757+Q,81
460 GOTO160
499 REM************************
500 PRINT"▣":PRINT"▩▩▩▩▩▩TTTTTTTGAME OVER"
505 IFD>HSTHEN HS=D
506 PRINT"▩TTTTPOANG";D
507 PRINT"▩TTTTHIGH SCORE";HS
510 GETA$:IFA$="S"THEN512
511 GOTO510
512 GOTO8
600 Q=Q-1:IFJ=1THEN D=D+1
610 J=0:GOTO150
700 Q=Q+1:J=1:GOTO150

READY.
```

ASSEMBLERSKOLAN

I förra numret lärde vi oss hur man tog in data till registren och la ut den igen. Vi tittade också på instruktioner som gav oss möjlighet att överföra data mellan olika register. Ännu har vi dock inte kommit så långt att vi kan konstruera några mer rutiner. Med den information vi skall titta på i detta nr så kommer vi dock att komma så långt att vi kan konstruera program som också gör något nyttigt. Det första vi skall göra är att gå igenom ALLA adressmoder som finns på 6502:ans (6510) repertoar så att vi har detta gjort en gång för alla.

6502:an anses ha den mest avancerade uppsättningen adressmoder. Detta har gjort att denna processor nog tillhör de svåraste att lära sig. I gengäld så blir det mycket lätt att lära sig en ny processor sedan man en gång lärt sig 6502 assembler. Detta gäller speciellt MOTOROLAS kraftfulla, och mycket populära, processor 6809 som i stort bara utvidgar och ger en större frihet i assemblerprogrammerandet.

För dig som glömt vad ADRESSMODER är för något så kan vi försöka oss på en definition.

Antag att vi vill ladda akumulatorn med ett tal, d v s med data. Vi kan göra detta genom att låta den aktuella datan ligga efter objektkoden d v s

LDA $3F

som laddar akumulatorn med $3F. Detta är en adressmod.

En annan möjlighet man kan tänka sig är att ladda akumulatorn med den data som finns lagrad i en minnesposition t ex

LDA $2000

som laddar akumulatorn med innehållet i minnespositionen $2000.

Vi har redan tidigare tittat på tre olika adressmoder nämligen

IMMEDIATE

Som verkar på det byte som ligger efter själva opkoden. Jmf LDA $3F ovan.

ABSOLUT

Som verkar på innehållet i en minnesposition d v s adressen till denna minnesposition anges efter opkoden. JMF LDA $C000 ovan.

ZERO PAGE

Vi har även tittat på detta specialfall som fungerar helt analogt med den ABSOLUTA adressmoden men med den skillnaden att den adress som anges skall vara en SIDA NOLL eller ZERO PAGE ADDRESS.

ABSOLUT,X

Detta är en s k INDEXERAD adressmod. Man använder här X-registret som ett OFFSET d v s ett tal som skall adderas till den adress man anger.

LDA $2000,X

kommer alltså att ladda akumulatorn med innehållet i minnespositionen vars adress erhålls genom att addera innehållet i X-registret med $2000. Om x-registret innehöll $10 när ovanstående instruktion kördes så skulle vi alltså laddat akumulatorn med INNEHÅLLET i minnesposition.

$2000 + $10 = $2010

Som Du ser så kan vi med en och sam-

ma instruktion nå vilket byte som helst i området.

$2000

$2000 + $F = S20FF

bara genom att ändra innehållet i X-registret. Observera dock att X-registret bara har 8-bitars längd och INTE 16-bitar som hos de flesta andra processorer. Detta är dock inte den stora nackdel som det kan synas vara till en början då det finns ännu kraftfullare instruktioner som kompenserar för detta.

ZERO-PAGE,X

är ett specialfall på ABSOLUT,X. Här skall alltså adressen vara en sida noll Adress.

LDA $CO,X

Laddar alltså akumulatorn med innehållet i den minnesposition som erhålls genom att addera.

$CO + (X)

(en parentes omkring ett register eller en adress brukar användas för att indikera INNEHÅLLET i registret eller adressen)

ABSOLUT,Y

Här använder vi innehållet i Y som OFFSEt istället för (X). För övrigt fungerar adressmoden helt analogt med ABSOLUT,X.

ZERO-PAGE,Y

Här använder vi innehållet i Y som OFFSEt istället för (X). För övrigt fungerar adressmoden helt analogt med ZERO-PAGE,X.

INDEXERAD INDIREKT

Detta är den minst använda adressmoden på hela 6502:ans repertoar och naturligtvis också den stökigaste. För att vi skall förstå hur den fungerar så tittar vi på ett exempel.

LDA ($CC,X)

kommer att ladda akumulatorn i den adress som erhålls om

$CC + (X)

adderas och vi sedan läser den adress som ligger lagrad här och laddar akumulatorn med innehållet i denna adress.

STÖKIG???????

Antag att

($CC) = $00
($C1) = $F0
($C2) = $19
($C3) = $FE
($C4) = $34
($C5) = $CE

och att

($F000) = $01
($FE19) = $02
($CE34) = $03

om (X) = 0 så kommer vi då att ladda akumulatorn med $01 ty

$CC + $00 = $CC

Denna position skall nu alltså innehålla en adress (på LÅG/HÖG format) som pekar ut den position som vi skall ladda akumulatorn från.

I vårt fall så innehåller

($CC) = $00
($C1) = $F0

Som alltså är adressen $F000 lagrad på LÅG/HÖG format. Innehållet i denna adress är $01 och alltså kommer akumulatorn att laddas med $01.

OBSERVERA att vi måste ha (X) som är jämna multiplar på 2 d v s 0,2,4,6,8,...

Om vi t ex testar

LDA ($CC,X)

med (X) = $01

så kommer vi att ladda akumulatorn med ($19F0) vilket med stor sannolikhet inte är det minnesinnehåll vi är intresserad av.

Vad laddas akumulatorn med i instruktionen ovan om (X) = $02 och (X) = $04?

Två ytterligare saker bör vi notera hos denna adressmod

1) Endast indexregister X kan användas.
2) Den adress som anges skall ligga på sida noll.

INDIREKT INDEXERAD

Detta är en MYCKET användbar adressmod som i mångt och mycket kompenserar för de trånga indexregistren. Dess assemblerform är i LDA fallet

LDA ($02),Y

Här laddas akumulatorn med innehållet i den adress som erhålls om man tar adressen som ligger lagrad på LÅG-/HÖG format med början på $02 plus innehållet i (Y) registret. D v s om

($02) = $2D
($03) = $20

och (X) = $10

så kommer alltså akumulatorn att laddas med innehållet i

$202D + $10 = $202D

Vi kan t ex tänka oss att vi har en tabell som börjar på adress $3000 där vi har lagrat 256 olika tal. Genom att lagra startadressen på LÅG/HÖG format på en sida noll adress t ex $02/$03 så kan vi genom att ha olika värden i Y-registret ladda ett valfritt element i denna tabell. Du bör naturligtvis vara medveten om att det även finns andra instruktioner än LDA som har denna adressmod.

När man talar om en adress som ligger lagrad i minnet så talar man ofta om en PEKARE eller VEKTOR. En pekare är en adress som pekar ut en speciell minnesposition medans en vektor är en adress om pekar på en assemblerrutin.

Slutligen så observerar vi att denna adressmod bara kan användas indexerad med Y-registret och att den pekare vi anger alltid skall ligga lagrad på sida noll.

ABSOLUT INDIREKT

Detta är en adressmod som bara finns för en enda instruktion JMP. Denna instruktion har samma funktion som BASIC:s GOTO med skillnaden att man i assembler hoppar till en adress istället för ett radnr. Den normala formen är den absoluta.

JMP $A000

som alltså innebär att programkörningen, EXEKVERINGEN, efter denna instruktion fortsätter på $A000. D v s dess funktion är helt enkelt att ladda PC (programräknaren) med den adress som anges efter opkoden.

Vid den indirekta moden så skall den adress vi vill hoppa till ligga lagrad i minnet, som vanligt på LÅG/HÖG format.

Om

($2000) = $DD
($2001) = $60

så innebär alltså instruktionen

JMP ($2000)

att programmet hoppar till $60DD.

Att låta olika rutiner börja med ett sådant här indirekt hopp brukar kallas PATCHAR. Normalt pekar den adress som ligger lagrad i minnet direkt på den kod som följer den indirekta hopp →

instruktionen. Men om man vill så kan man skriva om den rutin som skall köras och ändra VEKTORN till den nya rutinens startadress. På detta sätt så har fortfarande den aktuella rutinen samma adress.

Hos alla CBM maskiner är detta en teknik som används flitigt och man kan därför på dessa maskiner mycket enkelt ändra funktionen för IRQ, SAVE, PRINT eller någon annan rutin som normalt ligger lagrad i KERNAL (ROM).

Som vi sa så finns den indirekta adressmoden bara för JMP med ett litet trick så kan vi dock erhålla adressmoden för alla instruktioner som har INDEXERAD INDIREKT. Genom att låta (X) vara noll så erhåller vi här helt enkelt en indirekta mod med de enda kravet att adressen måste ligga lagrad på sida-noll.

IMPLIED

Detta är en verkligen enkel adressmod. Den har ingen operand alls d v s opkoden följs ej av någon data. TAX t ex som vi tittade på i förra artikeln har denna adressmod.

Här är det alltså enbart av opkoden klart var den data ligger som instruktionen skall verka på.

AKUMULATOR

Detta är helt enkelt ett specialfall av IMPLIED. Den enda skillnaden är att de instruktioner som har denna adressmod även har andra adresseringssätt till skillnad från dem som använder sig av IMPLIED som bara har en möjlig adressmod.

AKUMULATOR adresseringen innebär alltså att vi specificerar akumulatorn som det ställe där opkoden hittar den data den skall verka på.

RELATIV

Avslutningsvis skall vi titta på denna mycket viktiga adressmod.

6502:an har en grupp av instruktioner kallade BRANCH-instruktioner, som har denna adressmod som den enda. Vi skall i nästa nummer gå igenom dessa instruktioer en efter en men allmänt kan sägas att de testar en flagga i statusregistret och utför ett hopp om och endast om vissa villkor är uppfyllda som då består i att flaggan är satt eller nollställd.

Som Du inser så kan två saker hända:

a) Villkoret är uppfyllt! D v s ett hopp skall ske.

b) Villkoret är EJ uppfyllt! Programmet skall alltså fortsätta med insruktionen efter branch instruktionen.

Det hopp som sker görs ej genom att man anger den adress dit hoppet skall ske efter opkoden utan man låter istället ett byte ligga här.

Om detta byte är ett tvåkompliment så kan vi representera ett tal mellan —128—127 och genom att addera detta tal till PC så kan vi hoppa 128 steg bakåt eller 127 steg framåt. Det första vi måste förstå är alltså hur man representerar tal på tvåkomplimentform innan vi kan förstå hur detta går till.

TVÅKOMPLIMENT

Vi har i de inledande artiklarna i denna serie gått igenom talrepresentation hos datorer mycket grundligt men tvåkompliment representation har vi inte tittat på. Tvåkompliment representationen är helt enkelt ett sätt att representera negativa tal på binär form. Med ett byte kan vi ju representera ett tal mellan 0—255 d v s ha 256 olika bitkombinationer.

Om vi låter en bit i det binära talet tala om tecknet hos detsamma så kan vi alltså representera tal mellan.

—127—127

Om vi använder bit 7 för att indikera tecknet hos talet och låter denna vara satt om talet är negativt så blir alltså

%01111111 = 127

och

%11111111 = —127 (istället för 255)

Problemet med denna representation är bara att vi får två nollor

%00000000 = +0
%10000000 = —0 (istället för 128)

Ett annat sätt, som precis som ovan använder bit 7 för att indikera tecknet är ETT KOMPLIMENTFORMEN. Här byter vi helt enkelt varje satt bit hos den positiva motsvarigheten mot en nolla och tvärtom. D v s 85 som binärt har representationen

%01010101 = +85

har ett ETT KOMPLIMENT som har formen

%10101010 = —85

d v s är i absolut representation lika med 170. Vi kan nu om vi vill låta detta betyda —85 istället. Bit 7 indikerar även här att talet är negativt när den är satt. Problemet med de två nollorna kvarstår dock.

%00000000
%11111111

är båda representation av nollor.

För att komma ifrån detta så inför vi TVÅKOMPLIMENTFORMEN.

Ett tvåkompliment beräknar vi genom att

1. Ta ettkompliment av talet
2. Addera ett till resultatet från (1).

Vår reprsentation av —85 blir

%01010101 = 85
%10101010 ettkompliment
%10101010 + 1 = %10101011

Bit 7 kvarstår som teckenbit.

Vi har nu lyckats komma ifrån olägenheten med att ha två nollor för tar vi ettkompliment av noll så får vi

%11111111

och adderar vi ett till detta tal får vi noll igen.

Eftersom den extra nollan försvann så har vi fått ett större område. Med denna form kan vi representera tal mellan

—128—127

då ettkomplimentets negativa nolla %11111111 nu kan representeras av —1.

Antagligen tycker Du som de flesta andra att detta verkar jobbigt att beräkna då man hela tiden måste sitta och omvandla på bitnivå. Detta är dock inte nödvändigt.

Vill vi veta vilket tal som representerar —85 så subtraherar vi helt enkelt

256—85 = 171

och vi har direkt tvåkomplimentet.

%10101011 = 171

Forts på sid 55

I den tidigare artikeln om FORTH kunde du läsa om hur FORTH uppstod och hur det är jämfört med andra språk. Denna artikel ska istället presentera hur programmering i FORTH går till. Inga tidigare kunskaper behövs!

av Ola Johansson

OM FORTH – lär dig programmera

Som en liten repetition kan nämnas att FORTH tillkom på 1970-talet och dess grundare var en amerikan vid namn Charles H. Moore. FORTH finns att köpa till till båda VIC datorerna och kostar cirka 500:—. Jag rekomenderar boken ''Starting FORTH'' av Leo Brodie, Som är utmärkt för både nybörjare och för de som redan kan en del om datorer. Som framgår av titeln är boken skriven på engelska, så det är en fördel om du behärskar det språket.

Nu har det dock blivit dags att att fortsätta berättelsen om FORTH. Vi börjar med att titta på hur datorn tar hand om det ordet som du har givit den. Om det verkar för komplicerat kan du hoppa över det och ta det efteråt, när du har tittat på hur programmeringen går till.

Hur arbetar FORTH egentligen?

Till att börja med kan vi anta att du har matat in en rad av ord till datorn och tryckt på return knappen. Frågan är då vad datorn gör med detta. Jo, den försöker helt enkelt ''förstå'' det skrivna. Det gör den med hjälp av en del i språket som kallas för intrepretator. Den startar med att läsa in det första skrivna ordet, alltså fram till första mellanslaget. Sedan tar ett ord kallat EXECUTE över och försöker hitta ordet i ordlistan. Om det finner ordet så utförs de instruktioner som ordet står för. Om det inte skulle finnas ett sådant ord, så får ordet NUMBER i uppgift att översätta ordet till ett tal.

Skulle varken NUMBER eller EXECUTE hitta någon vettig betydelse i ordet så är det något fel. Detta meddelas genom att datorn skriver ut det felande ordet följt av ett frågetecken. Allt för att väcka eftertanke hos programmeraren. På samma sätt går datorn igenom ord för ord tills den kommer till slutet av raden. Då konstaterar den att den är klar för tillfället och talar om detta genom att skriva ut OK.

Datorn är nu klar för ett nytt ''uppdrag''. Sammanfattningsvis kan man säga att:

1) Datorn läser in ett ord.
2) Intepretatorn letar i ordlistan.
3) Om ordet hittas där anropas EXECUTE som utför ordet.
4) Ifall ordet inte hittas i ordlistan försöker NUMBER översätta det till ett tal.
5) Om varken NUMBER eller EXECUTE lyckas är det något fel och detta meddelas.

FORTH använder heltal

Jag nämnde i min förra artikel att FORTH använder heltal i sina beräkningar och att det kan ses som en nackdel av många. För att visa att man klarar sig utmärkt med heltal ska jag nu närmare utreda hur dessa fungerar och även tala om hur man kan simulera flyttal på olika sätt. Exempelvis kan jag redan nu beskriva ett enkelt sätt att få ''decimaltal'' med två decimaler. Man multiplicerar alla tal man matar in med hundra. Du skriver t ex in talet 17,42 som 1742 och när resultat skrivs ut får du själv korrigera talen så att de får rätt antal decimaler (om du inte skriver en funktion som gör det automatiskt förstås). Det verkar kanske för enkelt, men det fungerar faktiskt bra om du inte har för stora tal.

Det finns flera olika typer av heltal. Dels enkel precisionstal med eller utan tecken och dels dubbelprecisionstal, som kan fås med eller utan tecken. Enkelprecisionstal lagras i 16 bitar och dubbelprecisionstal i 32 bitar. Jag kan inte av utrymmesskäl gå in på vad detta innebär, men jag kan tala om att det röra sig om binära tal Fördelen med att inte tillåta tecken, utan endast ha positiva tal är att man kan använda den 16:e respektive 32:e biten som värdesiffra. Annars används den som tecken, på så sätt att en etta betyder negativ och en nolla betyder positiv. Den mest använda typen av heltal är de teckenförsedda enkelprecisionstalen och med dessa kan tal mellan −32768 och +32767 användas. Dessa gränser övervakas inte, så om du överstiger gränsen blir svaret helt enkelt fel. Inga felmedelanden skrivs ut.

För att räkna med dessa tal finns de vanliga räkneoperationerna + —* och /. Alla dessa ord kräver två tal på stacken (om du inte vet vad stacken är för något så kan du läsa min förra artikel. Annars får du fråga någon som har FORTH eller en HP-räknare) och ger som svar ett tal. Detta är antingen positivt eller negativt beroende på vilka tal du räknade med. Om någon av gränserna överskrids så meddelas inget om detta ,som sagt. Vid division kan tänkas att svaret ska bli ett decimaltal. Ska du dividera 21 med 5, som i FORTH skrivs 21 5 / , så får du svaret 4, trots att det borde vara 4,2.

→

→ För att få med bråkdelarna finns ett speciellt ord som heter /MOD . Detta ord ger dels svaret 4 och dels svaret 1, som ska tolkas som en femtedel. På detta sätt kan man utföra divisioner exakt trots att man ''bara'' har heltal. För att multiplicera ord med bråkdelar finns det ord som bl a kan användas för detta. Detta ord heter */ och kräver tre tal på stacken. Om du vill multiplicera 63 med 2/3 (för att få svaret 42) så skriver du 63 2 3 */. Fördelen med att använda detta ord istället för att beräkna 63*2 och sedan dividera med 3 är att mellanresultatet vid multipluikationen kan bli upp till 32 bitar ,trots att du bara räknar med 16. Fördelen är uppenbar om du vill beräkna 3/4 av 30000.

Stackmanipulering

Efter att ha tittat på hur heltalen fungerar ska vi nu se närmare på hur stacken fungerar. Eftersom det är på stacken som alla tal skall skrivas och resultaten hamnar, så är den en central del i FORTH. Man kan faktiskt säga att FORTH är en stack. För att kunna utnyttja stacken maximalt finns det fler aord som man kan använda om man vill ''trixa'' bland de tal som redan ligger där. För att exempelvis beräkna kvadraten av ett tal så behöver man två sådana tal, som man sedan ska multiplicera ihop. För att få två likadana tal använder man ordet DUP . Detta ordet duplicerar det översta talet på stacken. Det innebär att två likadana tal ligger överst respektive näst överst på stacken. Hur detta går till visas lätt med ett exempel: Vi definierar ett ord som beräknar kvadraten av ett givet tal. Definitionen blir:
KVADRAT DUP * ; OK
och ordet används på detta sätt:
4 KVADRAT . 16 OK

Som synes markerar understrykningarna vad du skriver och datorns svar kommer direkt efter, utan understrykning. Detta används för att skilja på kommandon och ansvar. Vilket underlättar läsningen av dessa. (Om du hoppade över första stycket är det lämpligt att läsa det nu.)

Förutom ordet DUP finns det flera ord som påverkar talen på stacken. Några av dem är SWAP, som byter plats på de två första talen, DROP, som tar bort översta talet och ROT, som roterar de tre översta talen enligt nedan:
före: 1 2 3
efter: 2 3 1
igen: 3 1 2
igen: 1 2 3

Efter tre ROT ser alltså stacken ut som tidigare. ROT kan vara användbart om man vill utföra någon beräkning på tre tal, utan att de försvinner från stacken. Då ''roterar'' man bara fram ett tal i taget. Utför beräkningen och roterar igen tills man har gått ett helt varv. För att kunna illustrerera de hittills presenterade orden har jag låtit nästa stycke bestå av exempel. Skrivsättet är detsamma som det jag tidigare har visat, dvs att det du skriver in är understruket och datorns svar kommer direkt efter, utan understrykningar. Ibland har kommentarer infogats inom parentes, vilket är det sätt man skriver kommentarer på i FORTH.

Några exempel

```
42 . 42 OK
4711 17 + . 4728 OK
21 5 / . 4 OK
21 5 /MOD . . 4 1 OK
." HEJ DU GLADE" HEJ DU GLADE
OK
: UPPHÖJT-TILL-TRE DUP DUP * * ;
OK
2 UPPHÖJT-TILL-TRE . 8 OK
17 42 . . 42 17 OK
17 42 SWAP . . 17 42 OK
: ÖKA-TRE-TAL-MED-FYRA 4 +
ROT 4 + ROT 4 + ROT ; OK
4711 42 17 ÖKA-TRE-TAL-MED-
FYRA . . . 21 46 4715 OK
: F(X) ( a b c — a+b*c) * + ; OK
(PARENTESEN OVAN VISAR DELS
HUR STACKEN SER UT) OK
(INNAN ORDET UTFÖRS OCH DELS
EFTER. DET ÄR PÅ) OK
(DETTA SÄTT MAN BRUKAR VISA
HUR ETT ORD PÅVERKAR) OK
(STACKEN) OK
1 2 3 F(X) . 7 OK
3 2 1 F(X) . 5 OK
```

Ytterligare ord

Innan jag visar fler exempel ska jag beskriva några ord som kan vara bra att ha. Först behöver vi ha en programslinga, som gör att datorn utför instruktioner flera gånger, även kallat en ''loop''. Denna ''loop'' inleds med ordet DO och avslutas med LOOP. Före DO ska man skriva två tal, nämligen programslingans start- och stoppvärde. För att plocka fram detta värde i loopen används ordet I som framgår av exemplen nedan. Andra lämpliga ord är några som sköter utmatning av text på bildskärmen. Vi har redan använt orden . (punkt) för tal och ." för text. De nya orden är CR SPACE, SPACES och EMIT . CR matar fram en ny rad, SPACE skriver ett mellanslag, SPACES skriver flera mellanslag och EMIT matar ut ett tecken. Ordet EMIT tar emot ett teckenvärde och skriver ut motsvarande tecken enligt en teckenstandard kallad ASCII.

Fler exempel

```
: RÄKNA 10 1 DO I . LOOP ; OK
RÄKNA 1 2 3 4 5 6 7 8 9 OK
: RÄKNA 11 1 DO I . LOOP ; RÄKNA
ISNT UNIQUE OK
RÄKNA 1 2 3 4 5 6 7 8 9 10 OK
: ABC 91 65 DO I EMIT LOOP ; OK
ABC ABCDEFGHIJKLMNOPQRS-
TUVWXYZOK :
NY-ABC 91 65 DO I EMIT SPACE
LOOP 3 SPACES ; OK
NY-ABC A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z OK
```

Om du har FORTH själv rekomenderar jag dig att experimentera friskt vid datorn. Risken att du förstör något är minimal. Enda möjligheten att ''kvadda'' datorn är att skriva med en hammare. Detta rekommenderas inte. När du skriver definitioner i FORTH så ska du försöka hålla dem så korta som möjligt och ge dem vettiga namn. Tänk på att en funktion får heta *vad som helst,* även ''$%&%$ om du skulle vilja det. Men som sagt, ge dem *vettiga* namn så att du förstår vad de gör bara genom att se på namnet.

Tips (och övning):
Skriv ord som byter färg på bilden och ramen runt om, samt något ord för att gå in och ur ''reverse-mode''. Du kan t ex kalla orden för BILD KANT och RVSON RVSOFF. I orden BILD och KANT kan det vara lämpligt att använda ordet C!, som placerar ett värde i en minnesadress (motsv POKE i BASIC).

Prova att använda ordet BILD och KANT i en programslinga så att bilden växlar färg snabbt. Dessa ord kan t ex se ut:

```
: BLIP 16 0 DO I KANT LOOP ; OK
: FLIMMER 0 DO BLIP LOOP ; OK
1 FLIMMER OK ''blippar'' en gång.
3 FLIMMER OK ''blippar'' tre gånger.
```

Nästa artikel kommer att handla om hur man kan få programslingor på flera sätt samt hur datorn kan välja mellan olika altenativ. Till dess ber jag dig som har FORTH hemma att prova olika definitioner i FORTH och se vad som händer. Var också nyfiken och skicka in frågor om FORTH till VIC rapport, så ska de besvaras i möjligaste mån.

Sammanfattning:

- FORTH tolkar din text med intepretatorn
- Intrepretatorn använder i sin tur orden EXECUTE och NUMBER
- Det finns flera typer av heltal
- De mest använda får vara mellan —32768 och +32767
- Det finns flera ord för att hantera stacken
- Skriv till VIC rapport och fråga om FORTH.

COMAL — BÄTTRE ÄN BASIC?

När nu äntligen COMAL finns att få till VIC-64 i diskladdad version med beteckningen 0.12, är det anledning att ta upp presentationen av detta effektiva och lätthanterliga språk i VIC-rapports spalter igen. Dessutom väntar vi ivrigt att få se produktionsleveranserna av den nya plug-in-modulen, som innehåller den hittills allra bästa av alla COMAL-80-varianter, CBM COMAL-80, version 2.00.

I COMAL kan man använda de två grundläggande reglerna för god programmering, som angetts av Kerninghan och Plauger i deras bok "The Elements Of Programming Style":

SÄG VAD DU MENAR, ENKELT OCH DIREKT.

VÄLJ VARIABELNAMN SOM KAN HÅLLAS ISÄR.

När Börge Christensen och Benedict Löfstedt byggde upp COMAL utgick man från den interaktiva miljö som BASIC hade och dess enkla kommandon, men man lade till från PASCAL hämtade strukturer, som gör det så lätt att utforma sitt program i avgränsade block. Dessa block kan sedan byggas samman till hela program.

Låt oss inte dröja utan hoppa direkt in i ett program, som kan visa oss fördelarna med COMAL framför BASIC.

Vi antar att en arbetsledare säger till en arbetare:

"Gräv sex hål för dessa staketstolper"

I ett formaliserat datorspråk kan detta omskrivas på två bestämda sätt:

a) REPEAT(=upprepa) sex gånger:
   GRÄV ETT HÅL FÖR STAKET-STOLPE
b) GRÄV ETT HÅL FÖR STAKET-STOLPE
   IF(=om) det är sista hålet THEN(=då)slutar du
   GOTO(=gå tillbaka till) FÖRSTA INSTRUKTIONEN

I många språk, inklusive de som fortsätter i FORTRAN/BASIC-traditionen får första alternativet följande utförande:

```
FOR räknare = 1 TO 6
gräv ett staketstolphål
NEXT räknare
```

Detta känns säkert igen av de flesta läsare som kan BASIC och FOR-slingan är en av de sex viktiga elementen i programmering. Men antag att arbetsledaren inte angav hur många hål som skulle grävas, utan bara sade:

"Gräv hål för staketstolpar tills du når fältets slut"

Motsvarande datorspråk skulle skilja sig litet:

a) REPEAT
   gräv ett stolphål
   UNTIL (=ända tills) du når slutet på fältet
b) gräv ett stolphål
   IF du nått fältets slut THEN stop
   GOTO första instuktionen

REPEAT/UNTIL-slingan är den andra huvudkonstruktionen vid programmering och här har visats två mycket använda sling-begrepp.

Men antag nu att vi vill ytterligare komplicera arbete och säga till vår arbetare att han skall gräva djupare hål om marken är lös. Instruktionerna skulle kunna ges på många sätt, men arbetsledaren kanske säger:

"Gräv stolphål till slutet av fältet.

Där marken är lös skall du gräva ett 60 cm djupt hål. Annars gräver du ett 30 cm hål."

Detta kan formuleras i FORTRAN-/BASIC-stil så här:

```
STIL a)
1. IF marken är lös THEN GOTO 4
2. gräv ett 30 cm hål
3. GOTO 5
4. gräv ett 60 cm hål
5. IF du är vid fältslut THEN STOP
6. GOTO 1
```

I Pascal/COMAL-stil skulle strukturen vara annorlunda:

```
STIL b)
REPEAT (=upprepa)
  IF marken är lös THEN
    gräv ett 60 cm hål
  ELSE(=i annat fall)
    gräv ett 30 cm hål
  UNTIL du är vid fälstslutet.
```

Visserligen kan man med vissa BASIC-varianter göra den första versionens struktur tydligare och mer läsbar. Men det står ändå klart redan av detta enkla exempel, hur tydligt språket i version b visar vad som händer i programmet.

Det kan kanske vara skäl att titta litet på de olika strukturer man talar om. Vi har tre mycket viktiga strukturer som används vid praktiskt taget all programmering:

Vi har redan behandlat dessa två:

REPEAT/UNTIL eller UPPREPA-/TILLS,

IF/THEN/ELSE eller OM-/DÅ/ANNARS

De tre andra är:

WHILE-slinga där operationerna upprepas så länge som ett villkor är sant och där man kontrollerar villkoret i början av strukturen.

CASE-strukturen där man har möjlighet till olika utgångar beroende på svaret på en fråga om CASE-satsens villkor. Denna struktur liknar BASICs "ON xx GOTO", men är kraftfullare, vilket kommer att visas i senare exempel i denna artikelserie.

Den tredje strukturen är PROCEDURER, på engelska PROCEDURES, d v s det som i BASIC/FORTRAN-terminologi kallas för subrutiner och som i BASIC anropas med GOSUB och radnummer, vilket kräver att man alltid håller noga reda på sina radnummer i de olika programavsnitten.

Det som gör COMAL så effektivt är kanske ändå allt man kan använda ända upp till 78 tecken långa namn och behöver inte längre kalla en variabel för kryptiska X3 eller ZP utan kan skriva "Mammas'bästa'recept'på'småkakor" om man vill. Varje tecken är signifikant, d v s tas med vid jämförelse mellan två variabelnamn — det betyder att "Mammas'bästa'recept'å'småkakor" är en helt annan variabel än den tidigare.

Vi kan jämföra BASIC med COMAL beträffande vilka strukturer och typfall som ingår i de båda språken så ser du fördelarna med COMAL:

	CBM BASIC	COMAL
SLINGOR		
REPEAT/UNTIL	NEJ	JA
WHILE	NEJ	JA
FOR/NEXT	JA	JA
VAL:		
IF/THEN/ELSE	DELVIS	JA
CASE	NEJ	JA
MODULER:		
PROCEDURES	DELVIS	JA
LÄSBARHET:		
IDENTIFIERARE	NEJ	JA

Som du ser ett ''ja'' och två ''delvis'' för BASIC mot hela sju ''ja'' för COMAL.

En ytterligare fördel med COMAL är att programmen automatiskt får indragning av programraderna efter de strukturer som används. Vi har försökt återge detta i exemplen med hålgrävningen ovan.

En annan fördel är automatisk radnumrering som ingår som en del i språket. Man behöver bara skriva AUTO och trycka på [RETURN] så dyker första radnumret upp på skärmen och markören står och väntar på att första programraden skrivs in. Dessa radnummer används för editering och för att hålla ordning på programradernas ordningsföljd, men utnyttjas INTE av programmet. Man kan alltså inte göra som i BASIC gå från ett programavsnitt till ett annat genom att skriva GOTO och ett radnummer!

En av de viktigaste anledningarna till att man så kraftigt propagerar för sttrukturerad programmering är just detta att man inte kan hoppa hit och dit i programet med GOTO-satser. Det har den mycket stora fördelen, att COMAL är lätt att läsa och lätt att förstå, lätt att underhålla och lätt att ändra i. Genom att använda omnumreringskommandot RENUM kan man dessutom återfå en snygg radnummerordning efter det att man har gjort tillägg eller strykningar — OCH MAN VET ATT INGA HOPPADRESSER HAR BERÖRTS!

Men låt oss gå vidare och titta på programmering i COMAL. Nedanstående första två program är ganska enkla men läsaren bör uppmärksammas hur användningen av långa, meningsfulla variabelnamn underlättar läsningen av programmen. Därtill komer möjligheten att få alla COMAL-ord med stora bokstäver och alla variabelnamn och andra identifierare med små bokstäver.

PROGRAM 1:

```
0010 FÖRSTA=35
0020 ANDRA=9
0030 TREDJE=FÖRSTA+ANDRA
0040 PRINT TREDJE
```

PROGRAM 2

```
0010 DIM SUBST$ OF 10, ADJ$ OF 10, VERB$ OF 10, MENING$ OF 20
0020 SUBST$:=" CALIGULA"
0030 ADJ$:="GRYM. "
0040 VERB$ :=" VAR "
0050 MENING$ :=SUBST$+VERB$+ADJ$
0060 PRINT MENING$
```

Det tredje programmet visar att COMAL har, precis som BASIC, FOR-slingan. Men lägg märke till hur effektiv indragningen av raderna gör läsningen och hur skrivningen med små/STORA bokstäver också underlättar.

*) OBS! COMAL omvandlar NEXT till ENDFOR och lägger till slingans identifierare när programmet körs med RUN.

PROGRAM 3

```
0010 FOR VECKA:=1 TO 4 DO
0020   PRINT "VECKA ";VECKA;"BORJAR"
0030   FOR DAG:=1 TO 7 DO
0040     PRINT DAG;
0050   ENDFOR DAG
0060   PRINT " ...VECKA";VECKA;"SLUTAR"
0070 ENDFOR VECKA
```

Utskrift efter RUN blir:
Vecka 1 börjar
1 2 3 3 4 5 6 7 ...Vecka 1 slutar

Vecka 2 börjar
1 2 3 3 4 5 6 7 ...Vecka 2 slutar

Vecka 3 börjar
1 2 3 3 4 5 6 7 ...Vecka 3 slutar

Vecka 4 börjar
1 2 3 3 4 5 6 7 ...Vecka 4 slutar

Det fjärde programmet illustrerar en REPEAT-slinga i vilken en fotbollsspelare springer med bollen ända till dess han råkar ur för en tackling, representerad av ett slumptal mindre än 0.2.

PROGRAM 4:

```
0010 REPEAT
0020   PRINT "TA ETT STEG."
0030 UNTIL RND(1)<.2
0035 PRINT RND(1)
0040 PRINT " DU FÖRLORAR BOLLEN."
```

En möjlig utskrift skulle kunna vara:
Ta ett steg
Ta ett steg
Ta ett steg
Ta ett steg
Ta ett steg
Ta ett steg
Spelet är slut.

Program nummer fem svarar på ett naturligt sätt på frågan: ''Hur många gånger måste talet 1 dubblas för att bli större än en mijon?''

PROGRAM 5:

```
0010 TAL:=1; MILJON :=1000000
0020 REPEAT
0030   TAL:=TAL*2
0040   RAKNARE:=RAKNARE+1
0050 UNTIL TAL>MILJON
0060 PRINT RAKNARE;" DUBBLERINGAR"
```

Det är ett välbekant faktum att en väldefinierad FOR-slinga bör ta emd i beräkningen att den kanske inte skall utföras alls. I t ex följande COMAL-program kommer ingen utskrift:

FOR k = 1 to 0
PRINT k
ENDFOR k

Å andra sidan måste REPEAT-slingan alltid utföras minst en gång. I sådan fal där detta inte är acceptabel och i andra fall där det verkar naturligast är WHILE-slingan nödvändig eller önskvärd. Fotbollsspelaren, som tacklas så snart han får bollen tar inga steg med den. Observera att vi i program 6 säger:

WHILE (skälet för genomgång av slinga)
-
-
-
ENDWHILE

Detta stämmer inte med program 5, där vi säger:

REPEAT
-
-
-
UNTIL (skäl för att avbryta = slingan)

PROGRAM 6:

```
0010 WHILE RND(1)<.1 DO
0020   PRINT "TA ETT STEG."
0030 ENDWHILE
0040 PRINT " NU FÖRLORADE DU BOLLEN!"
```

I program nummer sju visar vi hur proceduren BYT används i en s k ''bubble-sortering'' av tio färger i bokstavsordning. Läsare som känner till BASIC kommer att känna igen konstruktionen:

IF (villkor) THEN (satser)

Denna möjlighet finns också i COMAL, men den fullständiga satsen IF-/TEHN/ELSE ger värdefullare möjligheter.

PROGRAM 7 — BYTESPROCEDUR

```
0010 DIM FACK$(10) OF 10
0020 //
0030 FOR RAKNARE :=1 TO 10 DO
0040   READ FACK$(RAKNARE)
0050 ENDFOR RAKNARE
0060 //
0070 FOR KORNING :=1 TO 9 DO
0080   FOR PAR:=1 TO 10-KORNING DO
0090     IF FACK$(PAR)>FACK$(PAR+1) THEN BYT(FACK$(PAR),FACK$(PAR+1))
0100   ENDFOR PAR
0110 ENDFOR KORNING
0120 //
0130 FOR RAKNARE :=1 TO 10 DO
0140   PRINT TAB(5);FACK$(RAKNARE)
0150 ENDFOR RAKNARE
0160 //
0170 PROC BYT(REF STRANG1$,REF STRANG2$) CLOSED
0180   DIM TILLF$ OF 10
0190   TILLF$:=STRANG1$
0200   STRANG1$:=STRANG2$
0210   STRANG2$:=TILLF$
0220 ENDPROC BYT
0230 //
0240 DATA "ROD","BLA","GRÖN","ORANGE","VIOLETT","BRUN","LILA","MORKBRUN"
0250 DATA "AKVAMARIN","INDIGO"
```

Program åtta illustrerar hur man använder IF/THEN/ELSE-konstruktionerna. Det visar också hur ett program kan återge en naturlig ''uppifrån-och-neråt'' utförd analys med stegvis detaljökning. Det är lätt att se att om man vill räkna de olika slagen av tecken i en text så är det lättast att först skilja mellan bokstäver och icke-bokstäver. Detta utförs i huvudprogrammet och ytterligare detaljering sparas till nästa steg.

PROGRAM 8

```
0010 DIM TEXT$ OF 100, TEC$ OF 1
0020 TEXT$:="NI TALAR BRA LATIN"
0030 VOKALER:=0; KONSONANTER:=0; SIFFROR:=0; ANNAT:=0
0040 //
0050 FOR POANG:=1 TO LEN(TEXT$) DO
0060   TEC$:=TEXT$(POANG)
0070   IF TEC$>="A" AND TEC$<="Z" THEN
0080     BOKSTAV
0090   ELSE
0100     ICKE'BOKSTAV
0110   ENDIF
0120 ENDFOR POANG
0130 //
0140 PRINT VOKALER;"VOKALER"
0150 PRINT KONSONANTER;"KONSONANTER"
```

```
0160 PRINT SIFFROR;"SIFFROR"
0170 PRINT ANNAT;"ANDRA TECKEN"
0180 //******* SLUT PÅ HUVUDPROGRAMMET *      *****
0190 PROC BOKSTAV
0200  IF TEC$="A" OR TEC$="E" OR TEC$="I" OR TEC$="O" OR TEC$="U" THEN
0210   VOKALER:=VOKALER+1
0220  ELSE
0230   KONSONANTER:=KONSONANTER+1
0240  ENDIF
0250 ENDPROC BOKSTAV
0260 //
0270 PROC ICKE'BOKSTAV
0280  IF TEC$<>" " THEN
0290   IF TEC$>="0" AND TEC$<="9" THEN
0300    SIFFROR:=SIFFROR+1
0310   ELSE
0320    ANNAT:=ANNAT+1
0330   ENDIF
0340  ENDIF
0350 ENDPROC ICKE'BOKSTAV
```

Här har jag nu visat några inledande program, som visar hur väl COMAL fungerar som arbetsverkty, hur lätt detär att läsa och om du läsare skaffar dig en COMAL-diskett från någon Handic-återförsäljare och sätter igång själv kommer du att få stor glädje och nytta av din kunskap. I VIC-rapports spalter kommer att fortlöpande presenteras trevliga program och procedurer eller funktioner, som du kan använda i dina egna program, ändra och knyckla till så att de passar precis dina behov. Och det blir BÅDE lätt och roligt!

Ytterligare upplysningar om COMAL kan erhållas som medlem i COMAL-KLUBBEN I SVERIGE, c/o Åke Fredriksson, Gustavsbergsgatan 8, 431 37 Mölndal. Om du skriver, var snäll och bifoga svarsporto om du vill ha direkt svar.

Åke Fredriksson
Ordf. i COMAL-KLUBBEN I SVERIGE

Forts. från sid 48

VILLKORLIGA HOPP IGEN

Genom att ange ett tvåkompliment tal, som skall adderas till PC, efter BRANCH instruktionen så kan vi alltså hoppa 128 steg bakåt eller 127 steg framåt om det aktuella villkoret är uppfyllt.

De flesta maskinspråksmonitorer typ VICMON, 64MON eller liknande utför beräkningen av tvåkompliment talet och tillåter att man istället anger den adress dit man vill att hoppet skall ske. Detta gör det nödvändigt att behärska tekniken att beräkna detta tvåkompliment tal. För fullständighetens skull så visar vi dock två exempel på hur detta går till.

BEQ

är en instruktion som tillhör BRANCH-familjen. När processorn kör denna instruktion så tester den Z flaggan. Om Z flaggan är satt så adderas intruktionens andra byte, som är ett tvåkomplimenttal, till PC och programmet fortsätter att köra på den nya adress som PC nu pekar på. Om Z flaggan är nollställd så sker inget hopp utan programmet kör den instruktionen som ligger efter BEQ. Låt oss titta på ett program och gå igenom steg för steg vad som händer.

```
$C000   LDA $F000
$C003   STA $CC
$C005   LDA  $00
$C007   BEQ $FC
$C009   STA $CD
  o s v
```

Om Du tittar på adress $C005 i ovanstående program så ser Du att akumulatorn laddas omedelbart med $00 detta betyder att Z flaggan sätts här och alltså är satt när vi kommer till BEQ. Eftersom Z flaggan är satt så vet vi att BEQ kommer att göra ett hopp. Frågan är bara till vilken rad. $FC är lika med —4 och när processorn adderar $FC till PC så pekar PC på $C009. Adderar vi dessa båda så får vi den effektiva adressen $C005 d v s programmet kommer att hoppa tillbaka till adress $C005 och ladda in en nolla igen o s v. Om vi ändrar programmet till

```
$C000   LDA $F000
$C003   STA $CC
$C005   LDA  $FF
$C007   BEQ $FC
$C009   STA $CD
  o s v
```

så kommer Z flaggan INTE var satt när vi når BEQ och detta betyder att programmet kommer att fortsätta med

```
STA  $CD
```

När man beräknar det PC offset som skall ligga efter BRANCH instruktionen så räknar man alltså det antal steg som den nya adressen ligger på jmf med nästa opkods adress inte själva BRANCH-opkodens adress som man kanske skulle kunna vänta sig. Som sagt så är detta inte något man behöver bry sig om i normal fallet då Monitorn eller Assemblatorn sköter om det hela åt oss.

Kom alltså ihåg att offset skall beräknas från nästa instruktion inte från den adress som BRANCH instruktionen har.

Detta var allt för denna gång. I nästa nummer så skall vi gå genom samtliga BRANCH instruktioner på 6502:ans repertoar samt lära oss hur man programmerar s k LOOPar och liknande strukturer.

TIPPA 64

Johan bergkvist har gjort ett program som han kallar TIPPA 64. Ni som är nybörjare kommer säkert ha användning av detta. Programmet är inte speciellt unikt men ger en inblick i hur POKE-kommandon kan användas.

TIPPA 64 är till för VIC 64 och tar 11 bytes på en skiva (floppdisk). Bakgrunden och ramfärgen blir brun med ljusgrön text. Man ska försöka få ut ett tal genom att satsa på ett tal. Datorn ger besked om det är för litet eller för stort. Enkelt.

```
READY.

50 POKE 53280,9:POKE 53281,9
100 PRINT "♥"
150 PRINT "█"
170 PRINT "                    ********************"
200 PRINT "                    **** TIPPA 64 ****"
250 PRINT "                    ********************"
300 PRINT:PRINT      "         AV: JOHAN  BERGKVIST"
400 PRINT:PRINT
500 PRINT " DETTA SPEL GÅRUT PÅ ATT DU SKA"
600 PRINT " FöRSöKA KLURA UT VILKET TAL (1-250)"
700 PRINT " DATORN TÄNKER PÅ. DU SATSAR ETT TAL"
800 PRINT " OCH DATORN TALAR OM OM DET ÄR FöR"
900 PRINT " STORT ELLER FöR LITET."
1000 PRINT:PRINT:PRINT "  TRYCK 'P' FöR PRISLISTA."
1100 PRINT:PRINT "  TRYCK 'S' FöR START."
1200 GET A$:IF A$="" THEN 1200
1300 IF A$="S" THEN 3000
1400 IF A$="P" THEN 1550
1500 GOTO 1200
1550 PRINT "♥"
1600 PRINT:PRINT "                    PRISLISTA:"
1630 PRINT "                   ==========="
1650 PRINT
1700 PRINT " 1:A PRIS: RÄTT PÅ FöRSTA FöRSöKET":PRINT
1800 PRINT " 2:A PRIS: RÄTT PÅ 2:A - 5:E FöRSöKET":PRINT
1900 PRINT " 3:E PRIS: RÄTT PÅ 6:E - 10:E FöRSöKET":PRINT
2000 PRINT " 4:E PRIS: RÄTT PÅ 11:E - 20:E FöRSöKET"
2100 PRINT:PRINT:PRINT "  TRYCK 'I' FöR INSTRUKTIONER.":PRINT
2200 PRINT " TRYCK 'S' FöR START."
2300 GET B$:IF B$="" THEN 2300
2400 IF B$="I" THEN 100
2500 IF B$="S" THEN 3000
2600 GOTO 2300
3000 PRINT "♥"
3010 LET P=0
3100 LET I=INT(RND(1)*249)+1
3200 PRINT:PRINT "██  SATSA!!!"
3300 INPUT A
3320 PRINT:PRINT "  DU SATSADE PÅ:";A
3330 FOR Q=0 TO 1500:NEXT Q
3350 PRINT "♥"
3400 LET P=P+1
3500 IF A=I THEN 10000
3600 IF A<I THEN 5000
3700 IF A>I THEN 3800
3800 PRINT:PRINT "███   DIT TAL VAR FöR STORT."
3900 PRINT:PRINT "██   SATSA EN GÅNG TILL..."
4000 GOTO 3300
5000 PRINT:PRINT "███   DU SATSADE PÅ ETT FöR LITET"
```

```
5100 PRINT "{down}  TAL. SATSA IGEN..."
5200 GOTO 3300
10000 PRINT "{clr}"
10100 IF P=1 THEN 10107
10105 GOTO 11000
10107 FOR E=1 TO 200
10110 POKE 53280,6
10150 PRINT "{clr}"
10200 PRINT:PRINT "  DU FICK F ö R S T A P R I S !!!!"
10400 PRINT:PRINT "  DU HAR VERKLIGEN TUR !!!"
10500 PRINT:PRINT:PRINT " VARSAGOD !! FöRSTAPRISET: 10000 KR."
10520 POKE 53280,1
10530 FOR T=1 TO 20:NEXTT
10550 NEXT E
10570 POKE 53280,9:POKE 53281,9
10600 GOTO 30000
11000 PRINT "{clr}"
12100 IF P<6 THEN 12250
12200 GOTO 13000
12250 PRINT "{clr}"
12260 FOR E=1 TO 200
12270 POKE 53280,6
12280 PRINT "{clr}"
12300 PRINT:PRINT "  OJ!!! DU VANN 2:A PRISET: 5000 KR."
12400 PRINT:PRINT "  DU GISSADE RÄTT PÅ SATSNING NR";P
12500 PRINT:PRINT "  BRAVO !!!"
12510 POKE 53280,1
12520 FOR T=1 TO 10:NEXT T
12530 NEXT E
12550 POKE 53280,9
12600 GOTO 30000
13000 IF P<10 THEN 13300
13100 GOTO 13600
13300 PRINT:PRINT:PRINT "   DU TOG VÅRT 3:E PRIS: 1000 KR."
13400 PRINT:PRINT "  DU BEHöVDE";P;"SATSNINGAR"
13500 PRINT:PRINT "  GRATTIS !"
13550 FORT=1TO6000:NEXTT:GOTO 30000
13600 IF P<21 THEN 14000
13700 GOTO 15000
14000 PRINT "{clr}"
14100 PRINT:PRINT:PRINT "{down} DU FAR FJÄRDEPRIS:100 KR."
14200 PRINT "{down} GANSKA BRA..."
14250 PRINT "{down}{down}";P;"SATSNINGAR BEHöVDE DU."
14270 FORT=1TO6000:NEXTT
14300 GOTO 30000
15000 PRINT "{clr}"
15100 PRINT:PRINT:PRINT "  SYND. DU BLEV UTAN PRIS."
15200 PRINT:PRINT "  DU BEHöVDE";P;"SATSNINGAR."
15300 FORW=1TO2500:NEXTW
30000 PRINT "{clr}"
30100 PRINT:PRINT:PRINT "   VILL DU SPELA EN GANG TILL ? (J/N)"
30200 GET S$:IF S$="" THEN 30200
30300 IF S$="J" THEN GOTO 100
30400 PRINT "{clr}":PRINT "          H E J  D Å !!"
30500 PRINT "          ==================="
30600 FOR Q=1 TO 3000:NEXT Q
30700 PRINT"{clr}"
30800 END
```

MER OM COMAL

Vi börjar med en repetition av COMAL-strukturerna. De är i huvudsak samma som förekommer i Pascal och du kommer att förvånas över hur du tidigare klarat dig utan dem!

Första har vi villkorliga strukturer, där ett villkor avgör vad som skall ske!

IF...THEN...ELIF...ELSE...ENDIF
samt

CASE...OF...WHEN...
OTHERWISE...ENDCASE

I dessa båda satsstrukturer avgör utfallet av ett test av det i satsen angivna villkoret vilken utgång som programmet skall välja. Observera att varje struktur har en ingång och en utgång.

Så har vi villkorliga slingor, d v s upprepningssatser, där en procedur skall upprepas ett visst antal gånger, beroende på villkoret:

REPEAT...UNTIL

WHILE...DO...ENDWHILE

LOOP...EXIT WHEN...ENDLOOP
(Denna finns ej i version 0.12)

Till detta kommer en bestämd slinga, där antalet genomgångar är bestämt på förhand:

FOR...TO...STEP...DO...
NEXT(=ENDFOR)

Sist skall vi behandla de egentliga strukturmodulerna, funktioner och procedurer, där mycket kan ske inom respektive funktion eller procedur. Huvudskillnaden är att en funktion lämnar ifrån sig ett enda värde och en procedur kan kommunicera med flera variabler i ytterprogrammet.

Procedurer och funktioner kan vara öppna eller slutna och en viktig sak är att de variabelnamn som används innanför en procedur inte är kända utanför — det betyder att man kan använda samma namn i huvudprogrammet och i proceduren/funktionen utan att riskera sammanblandning!

Modulerna är 1) FUNKTIONER:

FUNC...REF...CLOSED...
IMPORT...RETURN...ENDFUNC

och PROCEDURER:

PROC...REF...CLOSED...
IMPORT...ENDPROC...EXEC

Detta var repetitionen, nu skall vi mer i detalj behandla varje struktur och börjar med IF-strukturen:

IF-strukturen

I CBM BASIC finns en mycket förenklad forma av IF-strukturen. COMAL utökar den till en flerradig struktur, som även omfattar en ELSE-del. Nu kan du bestämma ett villkor för utvärdering och om det är sant så kommer den första uppsättningen satser att utföras, men om det är falskt blir endast den andra uppsättningen satser (de som står efter ordet ELSE) att utföras. Det kan illustreras så här.

```
IF (villkor) THEN
  Satser som utförs om villkor är sant
ELSE
  Satser som utförs om villkor är
falskt
ENDIF
```

Ett exempel:

```
IF fel'räknare]0 THEN
  PRINT "Du genomförde denna övning med ";fel'räknare;"antal fel."
  PRINT "Dessa fel visar dig avsnitt som du bör öva mer på." ELSE
  PRINT "Fantastiskt, inga fel i denna övning!"
  PRINT "Det är inte många som har klarat denna övning så bra!"
  ENDIF
```

Indragning av satsblocken sker automatiskt när du skrivit LIST och tryckt [RETURN] och du ser hur de hjälper dig att klarare urskilja strukturerna. Strukturer kan kapslas i varandra och då kommer indragningarna att ange vilken inkapslingsnivå som gäller för strukturdelen. IF-strukturen ger också ett sätt att testa flera villkor inom samma struktur med nyckelordet ELIF (betyder ELSE IF). Du kan ha så många ELIF-delar som du behöver — eller inga alls. ELSE-delen kan också antingen tas med eller ej. Den fullständiga IF-strukturen visas nedan med ett exempel:

```
IF [villkor] THEN
  villkor 1—sant-satser
ELIF [villkor2] THEN
  villkor2 = sant-satser
ELIF [villkor3] THEN
  villkor3 = sant-satser
ELSE
  satser om något av villkor1-3 är
falskt
ENDIF
```

Ett programexempel:

```
IF noteringar[bank'besked THEN
  PRINT "Ditt bankbesked visar att du har"
  PRINT "mer pengar på banken än du själv noterat"
ELIF noteringar]bank'besked THEN
  PRINT "Du noterar mer pengar än bankens besked anger"
  PRINT "Kontrollera om du glömt skriva upp någon utbetalning"
ELSE
  PRINT "Dina noteringar stämmer med bankens"
ENDIF
```

CASE-strukturen

Ett flervägsval är möjligt i COMAL med CASE-strukturen. Det låter dig

lägga fram flera uppsättnigar satser, varje uppsättning kallad för ett "case", d v s ett "fall" för eventuellt utförande. Strukturen inleds med ett uttryck (antingen numerisk eller ett textuttryck) som skall utvärderas. Dess värde jämförs med de uttryck som är listade i början på varje fall. Om något uttryck stämmer överens kommer satserna i detta fall att utföras och resten av strukturen hoppas över. Om inget uttryck stämmer i något av fallen kommer de programsatser som finns efter OTHERWISE (= i annat fall) att utföras. Denna struktur ersätter den primitiva ON X GOTO [rad1], [rad2], [rad3]...

```
CASE [uttryck] OF
WHEN [uttryckslista 1]
  Satser 1

WHEN [uttryckslista 2]  Villkorliga
  Satser 2              avsnitt

OTHERWISE               Alternativt
  Otherwise-satser      avsnitt
ENDCASE
```

Ett exempel:

```
CASE val§ OF
WHEN "H","h","?"
  EXEC instruktioner
WHEN "A","a"
  CHAIN "ADDERA"
WHEN "S","s"
  CHAIN "SUBTRAHERA"
OTHERWISE
  PRINT "Jag förstår inte ditt val."
  PRINT "Du kan välja mellan:"
  PRINT " A = Addera"
  PRINT " S = Subtrahera"
  PRINT " H = Hjälp = instruktioner"
ENDCASE
```

Villkorliga sling-instruktioner

De flesta program har satser som skall utföras om och om igen. Det finns tre sätt att villkorligt hoppa ur sådana slingor: före inledningen av slingan, efter genomgång av slingan eller någonstans inuti slingan. COMAL har en speciell struktur som tillåter alla tre möjligheterna.

REPEAT-slingan utför första slingsatsen och kontrollerar därefter det villkor som specificeras i UNTIL-satsen. Slingan upprepas till dess villkoret utvärderas till sant (TRUE).

WHILE-slingan kontrollerar först villkoret. Om det utvärderas till falskt (FALSE) betraktas slingan som slutkörd och satserna överhoppas. Om det utvärderas till sant (TRUE) kommer satserna i WHILE-slingan att utföras och därefter kontrolleras villkoret om igen.

LOOP-strukturen fortsäter att utföra sina satser så länge som villkoret vid EXIT WHEN är falskt (FALSE). Denna konstruktion finns ej i version 0.12 utan endast i 2.00.

```
REPEAT
  Satser
UNTIL (villkor)

WHILE (villkor) DO
  Satser
ENDWHILE

LOOP
  Satser
EXIT WHEN (villkor)
  Satser
ENDLOOP
```

Några exempel:

```
REPEAT
  INPUT "Vad är svaret?: svar
  försök: + 1
UNTIL svar = rätt'svar OR försök]2

WHILE NOT EOF(infil)
  READ FILE infil: text$
  PRINT text$
ENDWHILE

LOOP
  READ FILE infil: namn$
EXIT WHEN namn$ = "*SLUT*"
  pekare: + 1
  vektor$(pekare): = namn$
ENDLOOP
```

Fast sling-struktur

Ofta vill man utföra ett helt block med satser ett bestämt antal gånger. COMAL har strukturer som liknar BASICs FOR...NEXT. COMAL omvandlar emellertid nyckelordet NEXT tiltl ENDFOR så att det står i samklang med övriga slingstrukturers avslutningsord. COMAL accepterar NEXT om man skriver det, men vid listning omvandlas detta av COMAL-tolken till ett ENDFOR såvida man inte genom att ge ett speciellt kommando behåller sitt NEXT.

Syntaxen är:

```
FOR
[styrvariabel]: = [startvärde]TO-
[slutvärde]STEP[stegvärde]DO
  Satser
ENDFOR[styrvariabel]
```

Delen "STEP[stegvärde]" är villkorlig och kan utelämnas. Om den ej anges, används stegvärdet 1 precis som i BASIC. Precis som i BASIC får du lov att skriva endast ENDFOR, utan att ange [styrvariabeln], den lägger emellertid COMAL-tolken till för dig. Observera att [styrvariabeln] jämförs med [slutvärdet] innan satserna utförs. Härigenom är det möjligt att hoppa över hela FOR-slingan helt och hållet. Till exempel kommer följande satser att överhoppas (alltså inte utföras ens en gång):

```
början: = 1; slut: = 0
FOR temp: = början TO slut
  PRINT "Nu är vi inne i FOR-slingan."
ENDFOR temp
```

Ett exempel på en slinga för läsning av 12 värden från datasatser och att lagra dem i en textvektor följer här:

```
FOR månad: = 1 TO 12 DO
  READ månads'namn$(månad)
  PRINT "Månad nummer";månad;"kallas";
  månads'namn$(månad)
ENDFOR månad
```

COMAL tillåter också en snabb enradig FOR-struktur som inte använder ENDFOR-avslutningen:

Syntax:

```
FOR[satyrvariabel]: = [startvärde]TO-
[slutvärde] STEP[steg] DO [satser]
```

Ett vanligt fall är en paus-slinga, som visas nedan (detta exempel visar också hur NULL-satsen kan användas, vilken, som namnet också anger, inte gör någonting):

```
FOR paus: = 1 TO 500 DO NULL
```

Procedurer och funktioner

Flerradiga procedurer och funktioner är en av COMAL's specialiteter. De är lätta att använda för nybörjaren och ger ändå möjligheter som bör tillfredsställa även den mest avancerade programmerare.

Det är lätt att skriva moduluppbyggda program i COMAL. Man kan när som helst anropa en procedur eller en funktion, var som helst i sitt program, helt enkelt genom att använda en EXEC-sats eller ett funktionsanrop. När ett program som körs möter en EXEC-sats, utför det denna procedur innan det fortsätter. Procedurer anropas med namn (kom ihåg, radnummer är inte intressanta för ett COMAL-program). Procedurer och funktioner tillåter parameteröverföring liksom användning av både lokala och globala →

variabler, d v s sådana som används både i huvudprogram och i procedur-/funktion. Och procedurerna eller funktionerna behöver nu inte ens ingå i programmet. De kan vara s k external, d v s finnas lagrade på disk för inlagring just där de behövs. På detta sätt har COMAL 2.0 praktiskt taget ett s k virtuellt minne, precis som stora datorer. Procedurer kan kapslas i varandra och ett program kan anropa ett annat. Eller t o m anropa sig själv, s k rekursion. COMAL tillåter användardefinierade nummeriska funktioner, heltalsfunktioner eller textfunktioner. Värdet på funktionen sänds tillbaka till huvudprogramet via RETURN-satsen (detta skiljer sig från RETURN-satsen i BASIC!). Även om här kommer att i huvudsak beskrivas procedurer så gäller att funktioner delar deras flexibilitet utom det att de anropas med ett funktionsanrop och sänder tillbaka ett värde, i stället för att anropas med en EXEC-sats.

Den viktigaste programraden för en procedur är första raden, som komer att kallas för huvudet. Här ger man proceduren dess namn, anger parametrarna om sådana finns och specificerar om den skall vara sluten (CLOSED) och ha lokala variabler eller inte. Version 2.00 ökar ut huvudet med en villkorlig external-indikator. På detta sätt kan PROC-satsen bli ganska komplex. Detta ger dig stor flexibilitet. Du behöver inte använda alla möjligheter, det räcker om du börjar med en enkel procedur.

Den enklaste PROC-satsen ger endast proceduren ett namn. Alla variabler fortsätter vara globala (= kända i hela programmet, huvudprogram och delprogram) och man använder inga parametrar:

```
PROC [procedur-namn]
```

T ex

```
PROC förfrågan
```

Man kan göra alla variabler inom proceduren lokala genom att lägga till ordet CLOSED (stängd) i slutet av procedurhuvudraden. En CLOSED procedur vet ingenting om de variabler som används i huvudprogrammet och programmet vet ingenting om de variabler som används inne i proceduren.

```
PROC [procedur-namn] CLOSED
```

T ex

```
PROC paus CLOSED
```

Men även en CLOSED procedur kan dela värden från huvudprogrammet genom parameteröverföring eller genom användning av IMPORT-satsen. De variabler som anges i parameterlistan för proceduren tilldelas startvärden som lämnas av den anropande EXEC-satsen. Enkel värdeöverföring är bara ”en-vägs” d v s in i proceduren. PROC-huvudet ser då ut så här:

```
PROC [procedur-namn] [parameter-lista] CLOSED
```

T ex

```
PROC etikett(namn$, adress$, stad$,postnr)CLOSED
```

COMAL låter dig också föra tillbaka värden till den anropande satsen. Då lägger man till nyckelordet REF före de parametrar, som man vill ha till ”två-vägs”-parametrar. Dessa variabler anropas därefter som ”referenser”, och används i stälet för de motsvarande variablerna i EXEC-satsen. Variablernamnet är fortfarande att betrakta som lokalt och kan inte komma i konflikt med någon variabel med samma namn i huvudprogrammet. Och den variabel som anropas som referens behöver inte vara samma som den i anropssatsen. Faktiskt brukar den vanligen vara olika. Som ett exempel, låt oss visa hur man kan mata in en text och därefter omvandla varje tecken som inte är en siffra till en punkt och varje siffra till bokstaven D:

```
DIM svar$ OF 80
INPUT"Skriv in någon text med siffror:"; svar$
EXEC omvandling(svar$)
PRINT "Efter omvandling ser det nu ut så här:";svar$
PROC omvandling(REF text$)CLOSED
  FOR tillf: = 1 TO LEN(text$) DO
    IF text$(tillf) IN "10123456789"
    THEN
      text$(tillf): = "D"
    ELSE
      text$(tillf): = "."

    ENDIF
  ENDFOR tillf
ENDPROC omvandling
```

Nu kan vi titta på en provkörning av detta program:

```
RUN
Skriv in någon text med siffror:
ABS123XYZ5
Efter omvandling ser det ut så här:
...DDD...D
```

Utöver att visa användningen av tvåvägs parameteröverföring, visar exemplet hur en del av en textsträng kan ändras utan att påverka resten av strängen. Mankan också se hur automatiskt strukturindragning av raderna ökar läsbarheten.

En CLOSED procedur kan också dela variabler med huvudprogrammet via IMPORT-satsen. IMPORT-satsen är i verkligheten en del av proceduren. En lista över variabelnamn hör till, och dessa kommer att delas med huvudprogrammet:

```
IMPORT namn$, betyg
```

Version 2.00 har utom all kraftfullhet och flexibilitet hos procedurer och funktioner även möjligheten att låta dem vara external till anropade program. Detta innebär att man kan ha alla sina mest använda procedurer lagrade på en diskett. Så länge man har denna diskett i sin diskettstation, kan alla program använda dess procedurer när som helst genom att helt enkelt skriva PROC-huvudet och avsluta med ordet EXTERNAL och korrekt filnamn:

```
PROC[procedurnamn] [parameter-lista] CLOSED EXTERNAL[filnamn]
```

T ex

```
PROC snabbsort(vänster,höger, REF namn$( )CLOSED EXTERNAL "snabbsort.e"
```

En external-procedur hämtas från disketten bara när den behövs. Och så snart som den är avslutad tas den bort ur minnet. Härigenom ger external-procedurer i verkligheten tillgång till ett virtuellt minnessystem, d v s ett dynamiskt minne som ändras efter behov. En annan fördel är att när man uppdaterar sin procedur, förbättrar den eller bara ändrar den, då behöver man bara göra detta en gång på sin huvud-biblioteks-diskett. Därefter kommer varje program som anropar procedurer att alltid ha senaste versionen av varje procedur. Och slutligen, programmen kan hållas mindre nu när procedurerna inte behöver skrivas in i varje program som använder dem. Detta kommer att tillåta fler program att lagras på en diskett.

Med external-procedurer och funktioner, program-sammankedjning, maskinspråks-länkning och avancerat felhanterings-system har COMAL utvecklats till att vara mer än ett språk enbart för nybörjar. Det är visst visserligen fortfarande ett mycket trevligt

språk för nybörjare, som senare kommer att finna att de kommer att kunna ha rik nytta av språkets många avancerade egenskaper.

Några andra avancerade egenskaper

Vissa av de avancerade egenskaperna som nu finns i COMAL inkluderar sådant som är hämtat från PASCAL och ADA men även sådant som är unikt för COMAL. En hel matris kan skrivas till diskett med en sats. Senare kan en matris med liknande dimensioner initialiseras med denna disk-fil, också med en programsats. Sekvensiella filer kan läsas och skrivas på två olika sätt. En metod är naturligtvis kompatibel med filer som skrivits i BASIC och låter dig få tillgång till filer från BASIC-program som du redan har. Den andra metoden använder en teckenräknare för textposter och medger att texten kan innehålla vilket som helst av ASCII-tecknen, inklusive de som vanligen utgör begränsningar vid BASIC.

COMAL har nu en "tids- och datum-stämplings"-möjlighet som är mycket användbar. Varje gång en fil lagras på disk så följer systemtid och datum med. På detta sätt kan du, om du har tre olika versioner av ett program på en diskett, men inte kommer ihåg vilket som är nyast, genom att läsa av tid- och datummärket för varje fil lätt se vilket som är det senaste. En utökad registerlistning kan nu också ta med tid och datum när varje fil lagrades på disketten. Backup-program (Säkerhetskopieprogram) kan också använda denna information och säkerhetskopiera endast de filer som har lagts till eller ändrats under den sensate veckan.

På så vis görs den veckovisa säkerhetskopieringen mycket lättare och kräver mindre tid och färre disketter.

Den avancerade IMPUT-möjligheten som COMAL nu har är anmärkningsvärd. Den ger fullständigt skydd från möjligheten att en användare "stökar till" en formatterad skärm under inmatningen, utan att du måste skriva invecklade procedurer för att styra inmatningen. INPUT-satsen accepterar nu inte tecken som självklart inte skall finnas med i någon inmatningsförfrågan. Det betyder bl a att markörflyttningar ignoreras, att "home markör" omdefinierats till att betyda "flytta markören till första tecknet i inmatningsfältet". Rensa skärmen har omdefinierats att utföra samma som "home markör", men det raderar också ut allt som skrivits in i inmatningsfältet och låter dig börja på nytt. COMAL tillåter inte heller användaren att hoppa in i citat-mode eller s k inskjutnings-mod under en INPUT-förfrågan. Likaså ignoreras omvändningskommando "Reverse Field".

Om du nu tycker att detta är fantastiskt, kommer COMAL att ytterligare utvidgas med INPUT AT-satsen. Denna sats tillåter dig att ange var på skärmen inmatningen skall börja och hur många tecken som kan tas emot som mest. Härigenom kan du med denna enda sats specifieras att inmatningen skall börja på rad 10 i läge (kolumn) 5 och inte kan vara mer än 6 tecken långt. Även en nybörjare kan nu producera några verkligen professionella "skott-säkra" program utan alltför mycken ansträngning.

Utöver de avancerade INPUT-möjligheterna ger COMAL också några avancerade OUTPUT (utmatnings)-möjligheter. Det inkluderar PRINT USING, vilket olyckligtvis aldrig implementerades (infördes i) CBM BASIC. Det tillåter också enkel skiftning från utmatning på skärmen till utmatning på printern (utmatning kan även dirigeras till en diskettstation). TAB och ZONE fungerar på samma sätt på både skärm och printer (det gör de inte i CBM BASIC!). Om du har en ASCII-printer (vilken icke-Commodore-printer som helst) kanske du redan upptäckt att PET ASCII inte är riktigt samma sak som standard ASCII. COMAL har möjlighet att automatiskt omvandla PET ASCII till standard ASCII för varje utmatning till skrivare. COMAL har också möjlighet att sända en radmatning med varje vagnretur. Vissa printrar kräver detta medan andra inte kan ha det.

Värdetilldelning till variabler är mer avancerat än CBM BASIC's. COMAL planerar ett kolon (:) före likhetstecknet i tilldelningssatser på samma sätt som PASCAL. Detta gör det lättare att skilja mellan tilldelning och jämförelse. Men man behöver inte skriva in kolonet, det lägger COMAL-tolken till själv. COMAL tillåter upp- och nerstigning av en variabel på samma sätt som CBM BASIC, men det finns också en ALGOL-liknande metod. Man behöver bara skriva ett kolon framför ett plus- eller minustecken och det blir en symbol för minskning eller ökning och är fullt jämförbar med kolon framför likhetstecknet. T ex om man vill minska sin kassa med en betalning kan man i BASIC skriva: KA = TA - BE

I COMAL kan detta skrivas:
KASSA :— BETALNING

Sträng-koppling (Concatenation) kan också ske med denna genväg:

BASIC: T$ = T$ + R$

COMAL: TEXT$:+ SVAR$

COMAL har ocksåTRUE (sant) och FALSE (falsk) som systemkonstanter. FALSE är alltid lika med 0 och TRUE lika med 1. När man använder en TRUE/FALSE-jämförelse är det mycket lättare att följa och mindre mångtydigt om du använder orden TRUE och FALSE än talen 0 och 1. COMAL har också några andra systemvariabler som är värdefulla. EOD sätts till TRUE när sista dataposten läses i datasatser. EOF sätts till TRUE när filslutet nås i en inmatnings-data-fil. COMAL har också en sats för att stänga av STOP-tangentens funktion och en systemvariabel, ESC, som då sätts till TRUE medan STOP-tangenten hålls nedtryckt. Du kan sedan använda STOP-tangenten till vilken funktion du vill i ditt program.

Därutöver har COMAL ett avancerat felhanteringssystem som kan användas i dina prograam. Nu, om ditt program möter ett fel under körning (t ex FILE NOT FOUND, d v s FILEN HITTAS INTE) kan programmet utföra lämpliga åtgärder i stället för att bara stoppa och skriva ett felmeddelande.

Begreppet "paket" från ADA har nu också lagts in i COMAL. Ett "paket" är ett slutet system komplett med startrutin och sina egna procedurer och funktioner, som kan vara globala eller privata (= lokala). Paketteckniken är en avancerad teknik och för externalprocedurerna ytterligare ett steg framåt. De flesta användare kommer inte att ta fram sina egna "paket", men det kommer så småningom att finnas många applikationsprogram, som använder dem eller kanske t o m programvaruhus som säljer "paket" precis som de nu säljer s k "tool kits" för BASIC.

Slutligen, om du använder en VIC-64, då har du både version 0.12 och småningom även 2.00 att tillgå med speciellt anpassade rutiner för att styra grafik-möjligheterna direkt från COMAL (d v s inga fler PEEKar och POKEar behövs). Detta innefattar också högupplösningsgrafik, s k Turtle-grafik (liknande den i LOGO) och sprite-styrnings-kommandon. Version 2.00 har också liknande kommandon för att styra ljudmöjligheterna med SID-chipet. Enbart dessa egenskaper borde ge tillräckliga skäl för att byta BASIC mot COMAL!

Frågor och svar om COMAL

Det har kommit åtskilliga frågor om COMAL och om möjligheterna att få COMAL till VIC-64 och vi publicerar här ett urval med frågor och svar:

Fråga: Finns några läroböcker om COMAL på svenska?

Svar: Ja, men Börge Christensens gamla bok som finns på Liber: COMAL I måste nog betraktas som föråldrad nu. Under sommaren kommer helt färska böcker från förlagsgruppen i Norrköping, både nyskrivna svenska och översättningar av kända böcker på danska och engelska, t ex COMAL HANDBOK av Lea Lindsay, COMAL-80 av Poul Östergaard, Strukurerad programmering med COMAL av Roy Atherton samt ytterligare någon bok. I dag finns en handbok till 0.12-versionen på VIC-64 klar i tryck och snart även 2.00-versionens 64-handbok, men de är inte egentligen läroböcker utan mer uppslagsböcker som presenterar de olika kommandona och nyckelorden.

Vilka böcker kan rekommenderas för en hemdatoranvändare?

Svar: BEGINNING COMAL av Börge Christensen är en nybörjarbok på engelska, som enligt vad som ryktas skall utkomma från Liber under sommaren. Även COMAL-80 av Poul Östergaard och Lindsays bok som både kommer från Förlagsgruppen, är bra nybörjarböcker. Lindsays Handbok är väl mer ett komplett lexikon över COMAL än en egentlig lärobok för nybörjare. I såväl VIC-rapport sm i COMAL-KLUBBENS tidning kommer instruktiva artiklr för att visa hur man kan utnyttja möjligheterna med COMAL.

Fråga: Hur kan jag köra COMAL på min VIC-64?

Svar: Skaffa dig 0.12-disketten redan nu! Den låter dig börja med en koncentrerad version av COMAL som kan ''mjukladdas'', d v s som inte kräver någon speciell maskinvara utan använder datorns minne för att lagra in programtolken. De program du utvecklar kan du sedan använda tillsammans med den fullt utbyggda versionen 2.00, som är aviserad till sommaren i form av en plug-in-modul som sticks in i användarporten på 64:an och låter dig ha hela 30K minnesutrymmen för program kvar. Med 0.12-versionen går en del minne åt för VOMAL-tolken och du får ca 10K över för program. Nu är detta ingen större fara eftersom COMAL låter dig använda skivminnet som extraminne och det gör att i praktiken kan du ha praktiskt taget obegränsat stora program i COMAL!

Fråga: Vilket diskettsystem används för COMAL?

Svar: Disketten för 0.12-versionen är en vanlig 5,25 tums som kan läsas i 1541-stationen (eller i en 4040-dubbelstation, om man har Superbox till sin VIC-64).

Fråga: Är disketterna kopieringsskyddade?

Svar: Ja, själva COMAL-tolken är kopieringsskyddad, men alla andra program, som också finns på disketten, kan fritt användas och kopieras, ändras eller ökas ut så att de lämpar sig utmärkt för den, som vill göra kvalificerade program och vill ha byggstenar att bygga vidare på.

Fråga: Finns det en kompilator för COMAL?

Svar: Alla versioner av COMAL innehåller också en s k RUN TIME COMPILER, d v s en kompilator som när programmet skall köras med ''RUN'' tar fram en kompilerad version av programmet. Den ger många av fördelarna med en vanlig kompilator, inklusive adresser till alla hopp och val i ett program. Vi har hört, att den faktiskt är snabbare än C-kompilatorn till VIC-64. COMAL-tolken fungerar så, att när man knappar in sin programrad och trycker på (RETURN) så kontrolleras först raden om den är syntatktisk korrekt och om alla nyckelord är rätt skrivna. Nästa steg sker när man ger kommandot ''RUN'' (eller ''SCAN'' i version 2.00). Då kontrolleras alla strukturer om de är korrekta och alla hopp ändras till absolut-adress-hopp. Därefter, när programmet faktiskt utförs, sker den sista delen av kompileringen — samtliga dessa steg är oerhört snabba och användaren märker ingenting annat än om han/hon gjort något fel. Då kommer tydliga felmeddelanden att upplysa om hur felet bör rättas — en ytterligare fördel med COMAL.

Fråga: Hur är kommandona för grafik och sprites på 64:an dokumenterade?

Svar: På COMAL-disketten finns en datafil, som ger syntaxen för kommandona, exempel samt en kort förklaring på de nära 50 nya kommandona. Under hösten 1984 kommer en speciell handbok, som behandlar grafik, sprites och ljud på 64:an. Den skrivs just nu av Len Lindsay i USA och skall översättas och utges av Förlagsgruppen i Norrköping.

Fråga: Kan BASIC-program köras i COMAL?

Svar: Nej, men om inte programmet är alltför ''spaghetti-betonat'' i BASIC kan du göra om det till ett COMAL-program. Annars kan du genom att knappa in ''BASIC'' och trycka på (RETURN) komma tillbaka till din BASIC-miljö. Men då måst du för at komma till COMAL på nytt, först stänga av datorn och därefter ladda in COMAL på nytt.

Fråga: Är COMAL-KLUBBEN I SVERIGE på något sätt knuten till Commodore, Datatronic, Handic eller Unicomal?

Svar: Nej, COMAL-KLUBBEN I SVERIGE är hel fri från alla bindningar och totalt oberoende av annat stöd än medlemmarnas. Företag kan ej vara aktiva medlemmar i klubben, endast stödjande medlemmar utan rösträtt.

Fråga: Min lokale handlare känner inte till COMAL, varför?

Svar: Av obekanta skäl har ej Datatro-

nic eller Handic satsat på detta nya kraftfulla språk i Sverige. Det är i stället Förlagsgruppen i Norrköping, som svarar för introduktionen hos handlarna och på skolorna. Man har förmodligen inte ännu täckt in samtliga handlare med sina presentationskurser och här kan du själv göra en insats genom att tala om vad du läst om COMAL och uppmana handlaren att ta kontakt med Förlagsgruppen för att få mer detaljerade informationer. Naturligtvis kan du också hänvisa till COMAL-KLUBBEN I SVERIGE, adressen står längst ned i denna frågespalt.

Fråga: Vilka skillnader är det mellan de olika CMB-versionerna av COMAL?

Svar: Version 0.12 har alla huvuddelar i COMAL-80-kärnan: programstrukturer med automatisk indragning av rader, skärmeditering, sammanknytning av program från disk, automatisk radnumrering, random-access-filer, två slags sekvensiella filer, programkedjning, flerradiga procedurer och funktioner med parameteröverföring, lokala variabler, avstängning av STOPP-tangenten, systemvariablerna EOD (End Of Data) och EOF (End Of File), inbyggd stränsökning, PEEK, POKE, SYS, omdedelbar syntaxkontroll, m m m m.

Version 0.12 finns för VIC-64 och för PET-datorer med upp till 32K minne. Version 2.00 kan köras dels i modul på 64:an, men den är inte leveransklar ännu i maj när detta skrivs och dels på alla PET-datorer med hjälp av ett speciellt COMAL-kort som fins att köpa från JAN EEN PROGRAMKONSULT i Surte (se listan nedan) och från COMAL-KLUBBEN. Dessutom finns en mjukladdad version på disk, som endast går att använda på Super-PET, d v s 8096 med 96K minne.

Version 2.00 är den fullständiga COMAL-80 med utvidgningar och möjlighet at köra program som utvecklats i version 0.12. I de utökade egenskaperna finns radnummeroberoende programkedjning från diskett, PRINT AT, PRINT USING, CURSOR, LIST av en procedur, utökade disk-kommandon. Både STORA och små bokstäver accepteras vid programinskrivning, man har GET$, KEY$, globala variabler i en CLOSED procedur, INTERRUPT-styrning av IEEE-avbrottsrutinen, RESTORE till en speciell LABEL, RENUMMER av endast en programsektion, SPC$, STR$, VAL och TIME. Version 2.00 har också egenskaper man väntar sig av ett stordatorspråk som: externa procedurer och funktioner, strukturerad felhantering, förbättrad filnamnsmetod, skyddade INPUT-satser, förmågan att enkelt och automatiskt LINKa relokerbar maskinkod till ett COMAL program och användardefinierade strängfunktioner.

Fråga: Hur är LOGO i förhållande till COMAL på 64:an.

Svar: 64:an har alla LOGO-kommandon för s k Turtle-styrning PLUS ytterligare några som tar tillvara 64:ans stora möjligheter. Se en separat artikel om grafik och sprites i COMAL, som publiceras i VIC-rapport snart.

Åke Fredriksson
Ordf. COMAL-KLUBBEN I SVERIGE

Adresser till leverantörer av COMAL:

FÖRLAGSGRUPPEN I NORRKÖPING
Nygatan 85, 602 34 NORRKÖPING
Telefon 011-13 40 80

Samtliga HANDICs återförsäljare
JAN EEN PROGRAMKONSULT HB
Trädgårdsgatan 8 A, 445 00 SURTE
Telefon 031-98 18 67

COMAL-KLUBBEN I SVERIGE
Kansli: c/o Åke Fredriksson,
Gustavsbergsgatan 8, 431 37 MÖLNDAL
Telefon 031-87 70 84

Maskinkodsmonitor till VIC 64

En BRA maskinkodsmonitor eller ännu hellre en assemblator är något som får oss "assemblerknuttar" att slappna av.

Någon tillräckligt bra assemblator har jag hittills inte sett på den svenska marknaden, men ryktet säger att det snart kommer att dyka upp en svensktillverkad dito.

En bra maskinkodsmonitor finns dock redan på marknaden men med den uppsjö av mer eller mindre fusktillverkade monitorer så är det lätt att köpa fel monitor, dvs en som är sämre men kostar mer.

Jag har under några månader haft den äran att jobba med 64-MON en maskinkodsmonitor för VIC-64 och tänkte att jag här på några rader skulle redovisa de erfarenheter jag har fått av detta ungänge. Jag anser att jag kan göra en rättvis bedömning av denna efter att ha arbetat och ibland sagt många fula saker om nära nog dussinet monitorer till CBM maskiner.

Vad har då 64-MON att bjuda oss fartdårar?

För att sammanfatta de möjligheter som monitorn ger oss så tittar vi på dess kommandon.

A
Assemblera en rad i minnet.

B
Med detta kommando kan man sätta TVÅ brytpunkter. De adresser man ger visas hela tiden överst på skärmen.

C
Jämför två minnesareor. (Denna finns dock ej dokumenterad i manualen. Dess SYNTAX är
C startadr1, slutadr1, startadr2.
De minnespositioner som skiljer sig åt mellan de två areorna listas sedan på skärmen.)

D
Disassemblerar ett minnesområde. Genom cursortangenterna kan man scrolla både bakåt och framåt i assemblerkoden.

F
Fyller ett minnesområde med ett värde.

G
Startar exekveringen av ett program i raltid.

H
Söker en sträng eller en byte kombination i ett område av minnet.

I
Listar ett område av minnet som ASCII kod. Med cursor tangenterna kan man scrolla både bakåt och framåt i minnet. Om man vill så kan man också lägga in ASCII kod i minnet med denna funktion.

L
Laddar ett program eller data från en yttre enhet.

M
Visar minnesinnehållet i hexadecimal form och i ASCII form. Man kan scrolla bakåt eller framåt i minnet med hjälp av cursortangenterna. Man kan dessutom lägga in både hextal och ASCII tecken i minnet med funktionen.

N
ändrar om alla absoluta referenser i ett program som flyttats från en area till en annan genom att addera ett ofsett. Man kan även använda funktionen på vektortabeller.

O
Utskriften sänds till skrivare istället för skärm.

Q
Programmet körs men break punkter söks och om sådan hittas går programmet in i WALK mod samt skriver ut register innehållet.

R
Visar processorns register.

S
Sparar ett program eller data på yttre enhet.

T
Flyttar ett minnesblock från ett område till ett annat.

V
Ger möjlighet att få det nuvarande innehållet i en minnesadress visad under WALK mod.

W
Stegar sig genom ett program och visar hela tiden assemblerkod och registerinnehåll. Detta är alltså WALK-mod.

X
återgång till BASIC.

£
Ger möjlighet att ge kommando till disk. Om bara £ skrivs ges statusen för disken.

*
Kopplar IN/UR BASIC/KERNAL ROM.

#
Konverterar hextal till decimala.

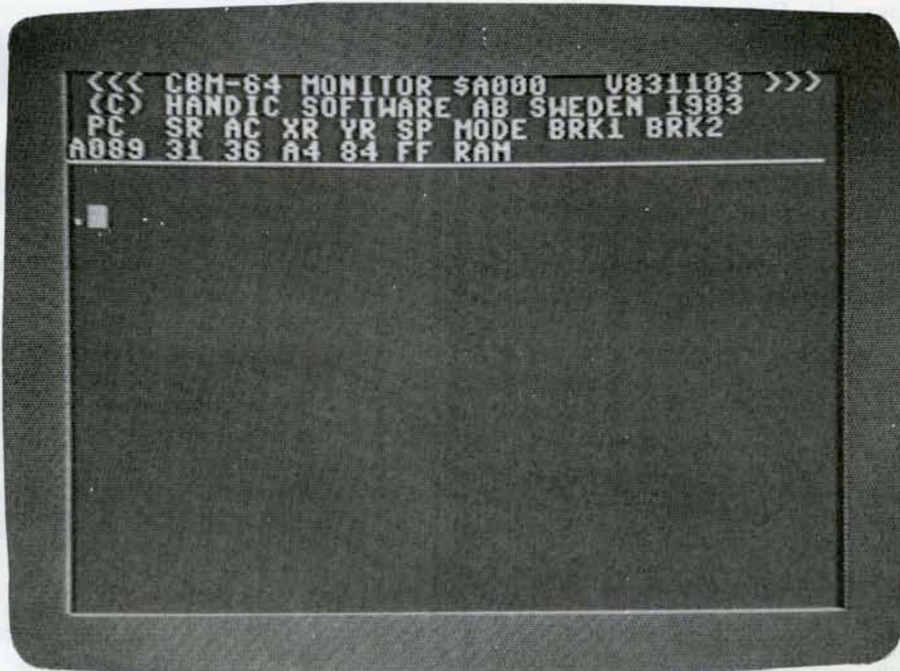
$
konverterar decimala tal till hextal.

Dessutom finns tre funktionstangenter definierade.

F1
Skriver ut den aktuella skärmen på skrivare.

F3
Ger möjlighet att ändra i processorns register.

F7
Visar en hjälpskärm som innehåller samtliga kommandon och deras syntax.

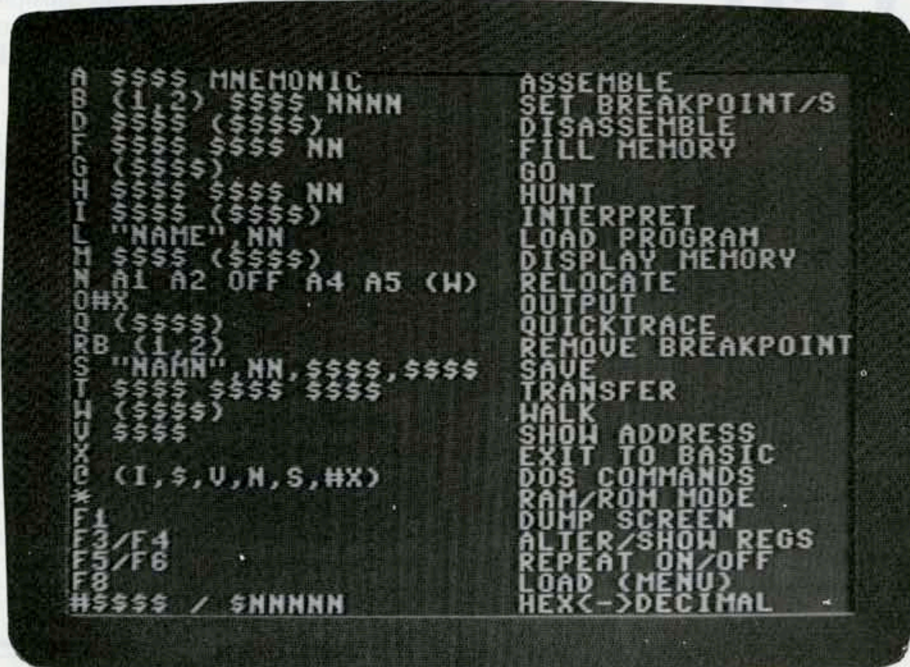


Det finns ingen anledning till att beskriva de kommandon som finns ytterligare. Istället så tar vi och går igenom de egenskaper som gör att 64-MON skiljer sig från andra monitorer på marknaden.

Det första vi måste nämna är att 64-MON kommer som PLUGG IN dvs en nätt liten smörgås som man pluggar in i datorn. På denna kassett så sitter en knapp märkt RESET. Ja, just det RESET. Med denna knapp så kan man alltså rädda många timmars arbete från att gå förlorat. ALLA har väl någon gång varit så självsäkra att vi testat vårt nykonstruerade assemblerprogram innan vi sparat det på disk eller band. Naturligtvis har det sagt POFF, ZAAAAP, BOOOOM och inget har fungerat längre. Till slut har man nödgats knäppa av och på datorn väl medveten om att det kära programmet för evigt är borta.

Tillsammans med 64-MON så kan man glömma detta. Ett tryck på RESET knappen och vips så är systemet igång igen utan att den kod man har lagrat försvunnit. UNDERBART!!!!! Att man tagit med denna knapp på kassetten tyder på att man tänkt på oss ”assemblernissar” men tyvärr har man glömt en sak. När man skriver kod i 6502 miljö så använder man nästan alltid sida noll. Reset rutinen hos 64-MON nollar dock hela sida noll så man kan få problem när man skall kontrollera vad som gick snett. Det viktiga kvarstår ju dock att man har sitt program intakt. Förutom RESET knappen så finns också två andra brytare i ett litet hål på ”lådan”. Dessa är märkta ”AUTO” och ”SYS 32768”. Enligt manualen så ger den ena ställningen av knapparna här AUTOSTART och den andra en uppstart av BASIC varifrån man själv får starta upp monitorn med SYS 32768.

I manualen står heller inget om hur man ändrar om de två små brytare som finns här. Det tog mig en hel dag att komma på att man skulle stoppa in en liten skruvmejsel i de små skåror som finns här och dra nedåt. Lyckan över att ha kommit på hur man ställde om dessa varade inte länge för i SYS läget kunde jag för allt i världen, hur jag än försökte, inte få monitorn att komma igång. Så fort jag skrev SYS 32768 så dök systemet. Av en ren tillfällighet råkade jag dock trycka RUN/STOP+RESTORE och vips så var jag inne i monitorn. Detta är antagligen tillrättat på senare modeller av 64-MON. Har Du dock en tidig modell så testa detta. Jag tänker inte klaga på manualen eftersom denna manual trots allt står långt över de manualer vi CBM-användare normalt har tillgång till. När monitorn startas upp så får man se en ganska udda skärm. Registerinnehåll och lite annat visas nämligen hela tiden på skärmens →



övre del. Detta är en underbar idé av tillverkaren och det är konstigt att ingen har kommit på det tidigare. GENIALT.

Som Du såg tidigare så fanns de normala kommandona men många av dessa är dock förbättrade och mycket användarvänliga. Låt oss titta på de av dem som har en funktion som skiljer sig från de tidigare monitortyper som sett dagens ljus.

A kommandot

Fungerar precis som vanligt. Till skillnad från många andra monitorer så kan man helt fritt ändra både data och memo ord utan att monitorn klagar. Detta underlättar handhavandet. Det enda som är lite väl långsökt är att man alltid måste ange "$" för ett hextal trots att bara hextal är tillåtna. Detta är ett förlegat tänkande som de flesta monitor tillverkare fortsätter med. För oss användare ger det en extra (SHIFTAD och tråkig) tangentnedtryckning på varje assemblerad innehållande en operand.

B kommandot

Här ges möjlighet att sätta två brytpunkter istället för en som hos de flesta monitorer. MYCKET BRA! Dessutom så visas de satta brytpunkterna på den dataarea vi nämnde ovan. ÄNNU BÄTTRE!

H kommandot

Möjligheten att söka en sträng utan att behöva koda den för hand till hexakod måste uppskattas. SUPERBT.

I kommandot

Här finns tyvärr en bugg i den version som jag har. När man en gång har gett I kommandot så kan man nämligen inte komma ur det. I manualen står det att STOP skall gå ur men detta fungerar ej. Man kan dock trycka på [F1] vilket fungerar alldeles utmärkt. Jag måste direkt säga att jag är överlycklig över detta kommando som förutom att ge mig möjlighet att titta på strängar i minnet tillåter att jag skriver in strängar som monitorn översätter till ASCII kod. Ett arbete som normalt har tagit mig åtskilligt med tid.

M kommandot

Har förutom den normala funktionen samma förträffliga funktion som I kommandot att man kan skriva in text. BRA BRA BRA.

O kommandot

Vem vill inte ha papperskopior av sin kod? Skööööönnnnt!

W kommandot

Register innehållet visas hela tiden precis som det skall vara. Hoppas att detta är klart för alla monitor tillverkare från och med nu.

£ kommandot

Detta kommando ger oss möjlighet att titta på directoryt (£$) eller formatera en diskett eller ge andra kommandon till diskenheten utan att störa det jobb man håller på med. Denna möjlighet lärde jag mig älska efter en minut tillsammans med 64-MON

[F1]

Att få en kopia av skärmen på skrivaren kan inte nog uppskattas.

[F7]

Man har alltid tillgång till en manual eftersom man få upp denna skärm när som helst. Man behöver därför inte bläddra så mycket i manualen. BRA!

[*]

Som kopplar ur ROM/RAM är en funktion som man tycker är nödvändig på 64:an då man gärna vill ändra de rutiner som finns här eller kanske lägga sin kod på den area som upptas av BASIC ROM:et dvs $A000-$BFFF. Tyvärr har jag blivit tvungen att konstatera att vissa kommandon inte fungerar ordentligt när man har kod på någon av areorna som delas av RAM/ROM. WALK kommandot tex blir helt värdelöst om koden råkar ligga på en av dessa areor. Detta är en stor miss som jag hoppas är tillrättad på senare versioner.

Möjligheten att konvertera mellan talbaser spar naturligtvis också mycket dyrbar tid och är något jag uppskattar. Det enda negativa man kan nämna i detta sammanhang är möjligheten att kunna göra enkla aritmetiska beräkningar i samband med konverteringen.

En annan bugg som kan nämnas är att den KERNAL rutin man anropar för att ta in ett tecken tecken från tangentbordet inte fungerar korrekt bla reagerar den inte på [F2]

Slutomdöme

Trots en del ganska tråkiga buggar så är detta den mest kraftfulla monitorn, för 64:an, på marknaden idag. Alla kommandon är väl genomtänkta och det är en njutning att jobba med den. Jag kan varmt rekommendera ett köp av denna monitor och kan garantera att Du INTE kommer att tycka att Du kastat dessa pengar i sjön. Antagligen har de flesta av de konstigheter jag beskrivit rättats till i de versioner som finns i handeln idag. Hur som helst är 64-MON det bästa köpet!

Hedman

Sprudlar du av ideér???

Var med i VIC rapports stora tävling!!!

Gör ett spelprogram

- Prova på att göra ett spelprogram.
- Lätt, svårt, spännande, roligt, alla spel tages emot med glädje. Skicka in ditt spel till redaktionen.
- OBS! Glöm inte att skriva om programmet är gjort för VIC 20 eller VIC 64

- 1:a pris. Det bästa spelet belönas med att du får välja ut valfria program från handic electronic till ett värde av totalt 500:—
- Dessutom belönas varje publicerat spel med valfria program från handic electronic till ett värde av totalt 300:—

Gör ett nyttoprogram

- Alla får vara med
- Dela med dig av dina kunskaper och skicka in ett nyttprogram till redaktionen.
- Glöm inte bort att skriva för vilken VIC programmet är gjort.

- 1:a pris — det vinnande nyttoprogrammet belönas med 1.000:—
- Dessutom belönas varje publicerat program med 400:—

Skicka in ditt program på en kassett eller en diskett tillsammans med en kort programbeskrivning.
Skriv namn och adress på programmet så att vi kan returnera det efter tävlingen.

Juryn för de två tävlingarna består av Nina Linander, Matts Nilsson och Elisabeth Höglund. Till vår hjälp har vi en mängd oberoende VIC-användare.
Juryns beslut kan inte överklagas.
Ev. beskattning av vinsterna får vinnarna själv ansvara för.

Frågor & Svar

brev från Karl Björklund som också handlade om The Hobbit, där han frågar om hur man kommer ut ur Goblins Dungeon. Svaret på detta kanske någon läsare kan ge honom då jag tycker att detta att " köra fast " är ett av The Hobbits goda förtjänster som Adventure program. Dessa program ska vara svåra och ger då en härlig känsla när man väl har löst hela äventyret och så att säga har gått i mål.

Keyboard till VIC 64
Jag är från Ludvika i Dalarna och har en VIC 64 sedan sju månader tillbaka och dessförinnan hade jag en VIC 20. Det mesta med VIC 64: an är bra men det finns för få möjligheter till musik. Det borde finnas ett keyboard som man kan koppla till den som man kan använda med ett program (ett snabbt i maskinkod förstås)och få en riktig synth. Följande egenskaper är önskvärda: 7—8 oktavers bredd, möjlighet att känna av flera tangenter samtidigt, 5—10 knappar för inprogrammerade effekter, 2—3 knappar för att byta tonområde, 5—10 knappar med komp, upp till fem möjligheter att byta ton. *OLA BLOMKVIST*

Ja, Ola vad du söker är en riktig super synth. Vi på redaktionen har frågat oss runt om detta. Din önskelista är lång och verkar svår att genomföra. För att få reda på om dessa möjligheter finns får vi vända oss till läsekretsen. Om ni läsare har någon idé så skriv till oss och berätta. Samtidigt som vi på red. ska fortsätta och söka på egen hand. Vem vet kanske kan vi tillsammans få iordning ett ordentligt keyboard.

The Hobbit
Jag är en ny VIC ägare och undrar om THE HOBBIT finns till VIC 20? Vad är priset och var kan man isåfall köpa det.
Jonas Holmsten i Sandviken
SVAR: Till VIC 20 finns inte The Hobbit på grund av att VIC 20's minne inte räcker till för detta program som är mycket stort och omfattande . Vi på red. fick ett

Hur kopierar man en flexskiva till en annan?
Bo Svensson vill veta hur man kopierar en flexskiva till en annan.

Detta kan göras på två sätt . Dels genom det kopieringsprogram som följer med diskdriven när man köper den men då krävs två diskdrivar, dels genom kopieringsprogram som är avsedda för en drivesanläggning. Dessa program arbetar på så sätt att de läser av så mycket av en flexskiva som rymms i minnet. Sedan byter man till den nya flexskivan och lagrar över den nya informationen för att därefter byta tillbaka till skiva ett och göra om proceduren tills all information är överförd till den nya skivan.

Toronto PET Users' Group

Daniel Ridings är medlem i världens största klubb för Commodore maskiner. Den heter Toronto Pet Users' Group. Man ska inte låta sig luras av namnet. De har stora programbibliotek för VIC 2O och VIC 64.

En av klubbens grundare är Jim Butterfield och han är ännu en flitig bidragare till klubbens programbibliotek, något som säger mycket om standarden på programmen. Det finns andra väl kända namn utöver honom.

Klubben har också mycket bra förbindelser med Commodore Canada. Commodore har satt hela sin utbildningsserie i "the Public domain" och skänkt hela serien till klubben, 51 packade disketter. Commodore har också lämnat ut en version av COMAL till "the Public domain". Alla intresserade kan få disken TPUG för $10 om man är medlem. Det finns en mängd program, uppåt 4000 allt som allt, som man kan få från dem. Här skriver jag mest om disk, men allt finns på kassett till ett något lägre pris, $6. Varje disk/kassett är full. I genomsnitt rör det sig om 35 till 40 program per disk.

Varje månad ger klubben ut en disk för PET , VIC 64 och VIC 20.

Det kostar $3O i årsavgift $4O om man vill ha programmen skickade med flyg. I avgiften ingår en tidning på 64 sidor som kommer ut tio ggr per år. Det finns artiklar alltifrån nybörjarnivå till avancerad maskinsspråksprogrammering. En klubb kan gå med och på det sättet få tillgång till programbiblioteket. Eller så kan var och en av medlemmarna ansluta sig till ett förmånligare pris och kanske få månadens disk till valfri maskin eller för flera maskiner beroende på hur många man är. Det är uttryckligen sagt att det är tillåtet för medlemmarna att kopiera programmen. Det är till och med möjligt för en butik att gå med att gå med i klubben och kopiera programmen för sina kunder, detta under förutsättning att man inte tar mer betalt än det som anses vara rimligt för den tiden man lägger ner på kopieringen.

Adressen är Toronto Pet Users' Group, 1912A Avenue Road Suite 1,Toronto Ontario M5M 4A1, Canada

Det finns en s k klubb som haft reklam i några svenska tidningar på sistone som kallar sig " Commodore Computer Club C 13 ". De säger sig ha massor av program. Jag har gått med och fått deras katalog. Ur det framgår att hela deras sortiment är hämtat ur TPUGs bibliotek men endast Pet program. Man bör varna för dem. De är dyrare än TPUG, de har ingen tidning och de levererar inte vad de lovar. Jag har beställt två disketter utan att fått dem eller fått någon förklaring och det är omöjligt att nå dem via deras telefonnr.

Jag hoppas att ordet sprider sig om TPUG. Alla som äger Commodore maskiner har glädje av det.

Om du är intresserad av mer information om TPUG så skriv till Daniel Ridings, Ryttareg. 1C, 415 O3 Göteborg



VIC 20

VIC 64

JELLY MONSTERS VIC-1905

1.	2497850	Peter Johansson, LYCKEBY
2.	1631860	Pelle Gustavsson, SUNDBYBERG
3.	705980	Anders Hellman, NOL

RAT RACE VIC-1909

1.	101520	Pär-Olof Håkansson, BJÄRRED
2.	96200	Olli Pesonen, ANGERED
3.	92000	Jocke Blyborg, STRÄNGNÄS

SUPERLANDER VIC-1907

1.	90160	Michael Larsson, SOLLENTUNA
2.	88000	Fredrik Gustafzon, GNESTA
3.	86000	Thomas Johansson, YSTAD

OMEGA RACE VIC-1924

(5 skepp)

1.	296400	Tomas Gordström, LINKÖPING
2.	274700	Fredrik Eklund, LINKÖPING
3.	163180	Jan Sjögren, ÅMÅL

JUPITER LANDER VIC-1907

1.	139900	Björn Lindman, HUDDINGE
2.	108200	Dan Andersson, SÖDRA SANDBY
3.	74900	Johan Harrysson, VRETA KLOSTER

CUP-FINAL (Commodore)

(Svårighetsgrad 9)

1.	5—0	Lars Kämpe, KVÄNUM
2.	4—0	Glenn Johansson, STENUNGSSUND
3.	0—2	Dr Darnio

GRIDRUNNER (LLamasoft)

(Level 1—)

1.	351820	Thomas Hartman, LULEÅ
2.	281920	Andreas Ruschkowski, GRÖDINGE
3.	260120	Jöran Omark, ÄLVSBYN

JUMPMAN (EPYX)

(Grand loop)

1.	54325	Peter Blomgren, HANDEN
2.	53025	Mattias Lind, SKELLEFTEÅ
3.	46750	Ola Hjalmarsson, UTANSJÖ

JUMPMAN JR (EPYX)

(Valfri hastighet)

1.	55825	Lennart Borg, TÄBY
2.	41050	Mats Palm, MÖLNDAL
3.	32475	STUFFE

DIG DUG (Atarisoft)

(Valfri start)

1.	742440	J. Aspengren, ATRIUM
2.	115020	Glen Johansson, STENUNGSSUND
3.	74860	Joakim Lundberg, LINDOME

SPELREKORD

Nu märks det att SPELREKORD har kommit igång ordentligt, det strömmar in rekord. Även flickor konkurrerar i denna maskulina "sport". I väntan på fler feminina rekord förbereds en "TJEJLIGA".

Ni får observera de krav som ställs under varje spelrubrik (meddela snarast vid felplacering-7.

SPELREDAKTÖREN
C.E.J.

VIC 20 ☐ VIC 64 ☐ Datum

Namn ....

Adress ....

Postnr .... Postadress ....

Spel ....

Erhållna poäng ....

Intygas av målsman eller annan myndig person.

Skrivet om VIC i amerikanska tidskrifter

Meningen med den här sidan är att recensera och tipsa läsarna om några av de bästa artiklarna för VIC i färska utländska tidskrifter. Det blir framför allt de ledande amerikanska tidskrifterna COMPUTE! och COMUPTE's Gazette för Commodore som kommer att behandlas. Vi hoppas att svenska VIC-ägare på detta sätt lättare ska kunna välja i den flod av tidskrifter och artiklar om mikrodatorer som nästan håller på att dränka oss.

av G. Berglund

COMPUTE's Gazette for Commodore, Maj 1984

VIC 64:ans resurser

Ljud på VICarna är temat i majnumret av "Gazetten". Framförallt 64:ans resurser behandlas. Den innehåller ju det mest avancerade ljudchip som finns i hemdatorer för närvarande. Tyvärr är det lite besvärligt att komma åt alla finesser som går att göra med 64:ans ljud eftersom man måste POKEa och PEEKa i det oändliga och känna till alla viktiga adresser i ljudchipet.

Det börjar nu komma programvara som underlättar det mödosamma arbetet med 64:ans ljud. I majnumret av "Gazetten" finns det bl a ett listat program för 64:an som kallas "Sound Sculptor" — ljudskulptören. Programmet arbetar med menyer och man använder joystick för att peka på det man vill göra. På detta sätt kan man enkelt välja det ljud man vill ha för var och en av de tre stämmorna. Förutom val av volym, vågform, attack/decay, sustain/release, och frekvens kan man också välja olika sätt att filtrera ljudet. Programmet tillåter dig också att spara de ljud du utvecklat på en tape eller disk. För den som vill använda ljuden i andra Basic-program beskrivs också ett sätt att, via SYS till maskinkod, komma åt de ljud som man utvecklat med hjälp av "Sound Sculptor".

VIC 20

För VIC 20 finns det tre artiklar som handlar om ljud. I en av dem visas hur man kan göra en rad effektljud, alltifrån spel av gräshoppor, via åska till ljud genom UFOs.

En annan artikel om ljud på VIC 20 innehåller ett maskinkodat program med vars hjälp man kan skriva en snutt musik genom att ge kommandot PRINT#1, "<Sträng>". Strängen kan innehålla alfabetets tecken från A till och med M och varje tecken motsvarar naturligtvis en ton. Tecknet Z producerar en paus i "tonsträngen".

Den tredje artikeln är en rätt så elementär genomgång av ljudmöjligheterna på VIC 20.

På spelavdelningen innehåller "Gazetten" i detta nummer tre listningar. "Props" för VIC 64, "Mind Boggle" för VIC 64 och 20 samt "Super Sprite" för VIC 64.

"Props"

"Props" är en guldgruva för den som vill lära sig lite mer avancerad spelprogrammering på 64:an. Här visar författaren i en utförlig beskrivning av programmet hur man blandar BASIC och maskinkod, sprites och avancerade ljudeffekter till en delikat anrättning. Spelet går ut på att med joystick lotsa en stackars pippi över en skärm fylld med snurrande propellrar till ett fågelbo där hans/hennes utvalda väntar.

"Mind Boggle"

"Mind Boggle" är en variant av det bekanta sällskapsspelet "Master Mind". De färger som man med logikens förmåga ska gissa sig fram till motsvaras också av vissa ljud som programmet producerar.

"Super Sprite"

"Super Sprite" är programmet som utnyttjar den möjlighet att utvidga sprites i X- eller Y-riktning som finns i VIC 64. Det hela går ut på att lotsa "Super Sprite" (en slags flygande "Superman") genom diverse kryptonitbarriärer på skärmen. En intressant teknik som visas i det här programmet är möjligheten att spara ett rekordresultat i programmet. På detta vis vet man att vid nästa tillfälle som man tar fram och använder detta spel, vilket bästa resultat som tidigare uppnåtts.

Utbildningsspel

Ytterligare en programlistning, nu i kategorin *utbildningsspel* som tycks vara så populära i USA, är "Ski Physics". Utbildningsspelen där manfår lära sig något vettigt samtidigt som man spelar ett kul spel kommer antagligen att få en enorm marknad allt eftersom skolorna datoriseras. Pass på pedagoger/programmerare! I "Sky Physics" lär man sig handskas med förhållandena mellan hastighet, avstånd och tid och samtidigt får man, som belöning vid rätt svar på uppgifterna, se en backhoppare göra ett flott skutt. "Ski Physics" finns listad både för VIC 64 och VIC 20.

"Gazetten" innehåller i varje nummer ett avsnitt ur en serie om maskinkodning för nybörjare. I detta nummer lär man ut bl a hur man i maskinkod flyttar ett objekt på skärmen.

Bland övriga artiklar i majnumret hittar vi också en som lär ut hur vår långsamma BASIC ska fås att gå lite snabbare. Som exempel bör man utelämna variabelnamnet efter NEXT i FOR...NEXT-loopar. Fler exempel på hur man får BASIC att snurra fortare hittar man i "Gazettens" majnummer.

COMPUTE!, Mars 1984

Tre listningar

I spelavdelningen i COMPUTE!'s marsnummer hittar vi tre listningar. Två av dem, "Roader" och "Barrier Battle", går till både VIC 20 och VIC 64. Det tredje, "Trident", passar endast till VIC 64:an.

"Roader"

"Roader" är som namnet antyder ett spel som går ut på att styra en bil på en mer eller mindre krokig väg. Vägen och bilen visas i fågelperspektiv och allt eftersom spelet fortgår blir vägen krokigare, hastigheten högre och en del hinder börjar uppträda på vägen. Det hela verkar inte särskilt originellt men eftersom listningen är ganska kort kanske det trots allt är värt mödan att knacka in programmet.

"Barrier Battle"

"Barrier Battle" kan beskrivas som ett "abstrakt grafikspel" för två spelare där det gäller att styra omkring med var sin barriär på skärmen och på så sätt innesluta sin motspelare. Idén verkar originell men eftersom spelet saknar en realistisk "story" så blir det kanske svårt att känna sig riktigt "delaktig" som spelare.

"Trident"

"Trident", som passar till 64:an, verkar däremot nästan otäckt realistisk och är dessutom helt i maskinkod (flera sidor med siffror som ska knappas in). Det hela utspelar sig ombord på utbåten "Trident" där uppgiften är att, med hjälp av utbåtens dator och ens egna snabba beslut, förgöra inkommande fienderaketer. Det är alltså ett slags krigssimuleringsspel där ens enda kontakt med fienden är den information man får av sin dataskärm ombord på ubåten. Som hjälp för att få inknappningen av alla siffror rätt finns ett hjälpprogram, MLX, som måste knappas in före man börjar inknappningen av huvudprogrammet som består av bortemot sextusen siffror.

"64 Explorer"

En återkommande spalt i COMPUTE! är "64 Explorer" där man får diverse tips om 64:an. I detta nummer bl a tips på hur man kan fixa en "reset-knapp" så att man kan klara sig ur besvärliga situationer då maskinen "hängt sig" utan att behöva stänga av den. På så sätt kan man behålla värdefullt minnesinnehåll.

Bland artiklar om programmeringsteknik hittar man bl a en som handlar om "Relational Operators". Om man någon gång i ett program stött på en rad av typen J = J * (J<10) och undrat vad det är som försiggår så kan man hämta en del information i artikeln om "Relational Operators".



"Random Music"

En ljudartikel för båda VICarna i COMPUTE! är "Random Music". Som titeln antyder är det ett program som spelar slumpmässig "musik". Ett kul experiment värt att pröva.

Var kan jag finna tidskrifterna?

Hur får man tag i dessa tidskrifter? I välsorterade Pressbyråkiosker bör man kunna hitta åtminstone COMPUTE!. COMPUTE!'s Gazette for Commodore kan bli lite knepigare att få tag i, vilket är synd eftersom det är en av de bästa tidskriften för VIC-ägare. Ett sätt att få tag i "Gazetten" är att tala med en välsorterad Pressbyråkiosk och be den ta hem ett exemplar till just dig. Ett annat sätt är att gå till en välsorterad bokhandel och be den hjälpa dig med en prenumeration. Man kan också ordna prenumerationen själv, men då måste man först ordna en internationell check (money order) utställd på det förlag där tidskriften ges ut. Detta gör man på sin bank. Därefter skickar man checken i ett brev tillsammans med sitt namn och adress samt önskemål om prenumeration till förlaget. Här nedan fins adresser och prenumerationspriser för de aktuella tidskrifterna.

COMPUTE!'s Gazette for Commodore
Subscription Services Department
P.O. Box 961
Farmingdale, N.Y. 11737
USA

Prenumerationspriset per år för COMPUTE!'s Gazette är US$25 (surface mail) eller US$48 (air mail.-7 "Air mail" får man tidskriften en månad tidigare.

COMPUTE! Magazine
P.O. Box 914
Farmingdale, N.Y. 11737
USA

Årsprenumerationen på COMPUTE! kostar US$30 (surface mail) eller US$42 (air mail). ●

Sänka skepp VIC 20

Sänka skepp är ett gammalt hederligt spel. Robert Pallbo har här nedan gjort ett spelprogram för VIC 20 som just bygger på detta spel.

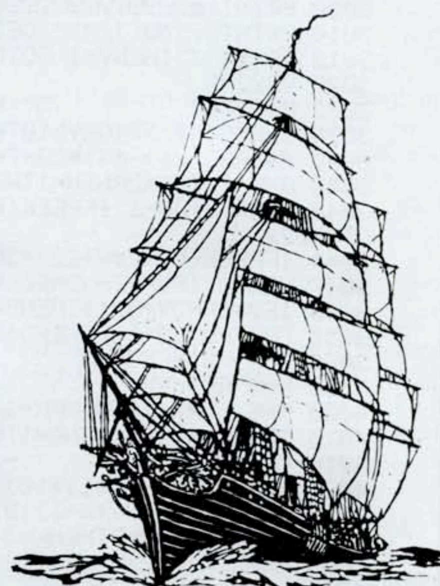
Spelet är uppdelat i två program för att få plats i en oexpanderad VIC 20.
Program 1: Grafik och instruktioner.
Program 2: Själva spelet.

Reglerna är desamma som till vanliga sänka skepp. Å, Ä och Ö finns med eftersom det är svensk text. Man spelar mot datorn och själv tycker jag att det är en jämn kamp. Programförklaring finner ni längst ner.

Programförklaring

100—135 Ritar ut spelplanen och hoppar till 3000 och 4000.
150—250 Din bombning.
3000—3500 Du sätter ut dina skepp.
4000—4050 Datorn sätter ut sina skepp.
5000-5001 Ljud för träff.
5010-5020 Ljud för bom.
5050— 5070 Någon har vunnit. Ljudsignal. Skriver ut vem. Frågar om man vill spela igen.
5100—5150 Datorn gissar.
5200—5310 Datorn håller på att sänka ett skepp som den har hittat på raderna 5100—5150.

Om någon inte orkar slå in programmet kan ni få en kassetkopia för 30 kr. Ring till Robert Pallbo 0451-515 68



```
READY.

10 REM ***************
20 REM    AV R.PALLBO
30 REM       1984
40 REM ***************
50 POKE52,28:POKE56,28:POKE36869,255:FORI=0TO511:POKE7168+I,PEEK(32768+I):NEXT
60 POKE36879,25:PRINT"    **SÄNKA SKEPP**"
70 PRINT"      AV R.P 1984."
80 PRINT"***********************"
90 PRINT"    INSTRUKTIONER"
100 PRINT"PLACERA UT DINA SKEPP":PRINT"GENOM ATT ANGE VÅGRÄT":PRINT"ELLE
R";
110 PRINT" LODRÄT.":PRINT"  ANGE VÄNSTRA RESP.":PRINT"ÖVERSTA POSITIONEN
."
120 PRINT"  DU HAR 5 SKEPP.":PRINT"TRYCK!"
130 FORR=7168TO7168+6*8-1:READA:POKER,A:NEXT
140 DATA255,129,129,129,129,129,129,255,255,129,129,129,129,129,129,255
150 DATA255,195,165,153,153,165,195,255,126,255,255,231,231,255,255,126
160 DATA0,38,105,169,41,41,41,38,130,135,133,255,191,158,140,255
170 FORR=7384TO7407:READA:POKER,A:NEXT
180 DATA36,24,36,66,126,66,66,66,36,24,36,66,66,66,36,24
190 DATA24,0,24,36,66,126,66,66
200 GETA$:IFA$<>""THENPRINT"B - TRÄFF":PRINT"C - BOM":GOTO220
210 GOTO200
220 PRINT"SPELAR IN SPELET..."
230 FORR=1TO8:POKE630+R,ASC(MID$("LOAD RUN",R,1)):NEXT:POKE635,13:POKE639,13:POK
E198,9

READY.
```

```
110 POKE36869,255:PRINT"123456789D":FORR=1TO10:PRINT""CHR$(64+R)"AAAAAAAA
AA"
130 NEXT:PRINT"123456789D":FORR=1TO9:PRINT""CHR$(64+R)"AAAAAAAAAA":NEXT
135 PRINT"JAAAAAAAAAA";"":G=6:GOSUB3000:GOSUB4000:CLR:L=36878:S=L-3:PRINT"";
150 FORR=1TO15:PRINT,"                ";:NEXT:B$="DU"
160 PRINT"","VAR BOMBAR":PRINT,"DU":PRINT,;:INPUT"";A$,Y$:Y=VAL(Y$):Y$=""
170 X=ASC(A$)-65:PO=7966+X*22+Y:IFPEEK(PO)=0THENPRINT""," TRÄFF!!! ":GOSUB5000:
GOTO200
180 PRINT""," BOOOM!!  ":GOSUB5010:GOTO250
200 POKEPO,2:POKEPO+30720,2:TR=TR+1:IFTR=17THEN5050
210 FORR=1TO500:NEXT:GOTO150
250 POKEPO,3:POKEPO+30720,0:FORR=1TO2000:NEXT:GOSUB5100:FORT=1TO500:NEXT:GOTO150
```

```
3000 PRINT"■▒▒▒▒▒▒▒▒▒▒▒▒▒▒SÄ▒TT UT▒■■■■■■■■■▒▒DINA SKEPP!":PRINT,"▒VAGRAT▒▒▒",,;
3010 PRINT,,"▒▒(J/N)":GETX$:IFX$<>"J"ANDX$<>"N"THENPRINT"▒▒",:GOTO3010
3012 IFX$="J"THENV=1:GOTO3022
3020 V=22
3022 PRINT,"              ▒":R=-G:PRINT,"■▒POS.       ■■■■■";;:INPUTX$,Y
3030 IFASC(X$)>74ORY>10THEN3022
3040 X=(ASC(X$)-64)*22+7680:PO=Y+X:IFG=3ANDN=0THENN=1
3042 G=G-1:IFG=2ANDN=1THENG=3:N=2
3043 FORR=0TOG-1:IFPEEK(PO+R*V)=5ORPEEK(PO+R*V-1)=5ORPEEK(PO+R*V+2)=5THENG=G+1:G
OTO3022
3044 IFPEEK(PO+R*V+22)=5ORPEEK(PO+R*V-22)=5THENG=G+1:GOTO3022
3047 NEXTR:IFV=22ANDASC(X$)-64+G>11THENG=G+1:GOTO3022
3048 IFV=1ANDY+G>11THENG=G+1:GOTO3022
3055 FORR=0TOG-1:POKEPO+R*V,5:POKEPO+R*V+30720,G:NEXT
3060 IFG>=3THEN3000
3500 N=0:RETURN
4000 K=6:FORW=1TO5:K=K-1
4015 IFRND(1)>.5THENC=1:GOTO4020
4016 C=22
4020 X(W)=INT(RND(1)*10)+1:Y(W)=INT(RND(1)*9)+1:IFC=22ANDX(W)+K>10THEN4020
4023 IFC=1ANDY(W)+K-1>10THEN4020
4025 IFK=2ANDK3=0THENK=3:K3=1
4030 PD=X(W)*22+Y(W)+7944
4035 FORR=0TOK-1:IFPEEK(PD+R*C)=0ORPEEK(PD+1+R*C)=0ORPEEK(PD-1+R*C)=0THENGOTO402
0
4036 IFPEEK(PD+R*C+22)=0ORPEEK(PD+R*C-22)=0THENGOTO4020
4040 NEXT:FORR=0TOK-1:POKEPD+R*C,0:NEXT
4050 NEXT:RETURN
5000 POKEL,15:FORR=1TO4:FORE=200TO255STEP.5:POKES,E:NEXT:POKES,0:FORE=1TO10:NEXT
:NEXT
5001 POKEL,0:RETURN
5010 POKEL,10:FORR=200TO128STEP-2:POKEL-1,R:POKES,R:NEXTR:POKEL-1,0:POKES,0:POKE
L,0
5020 RETURN
5050 POKEL,15:FORR=1TO5:FORE=128TO255STEP2:POKES,E:NEXT:POKES,0:FORT=1TO10:NEXT:
NEXT
5055 POKEL,0:PRINT"▒"
5060 PRINT"▒***";B$;"▒ VANN****":PRINT"▒▒▒▒▒▒VILL DU SPELA IGEN J/N":GETF$
5065 IFF$="J"THENRUN
5067 IFF$="N"THENEND
5070 PRINT"▒▒▒▒▒";:GOTO5060
5100 IFQ<>0THEN5200
5120 X=INT(RND(1)*10)+1:Y=INT(RND(1)*10)+1:PD=7680+Y*22+X
5125 IFPEEK(PD+1)=2ORPEEK(PD-1)=2ORPEEK(PD+22)=2ORPEEK(PD-22)=2THEN5120
5127 IFPEEK(PD)=1AND(PEEK(PD+22)=3ORPEEK(PD-22)=3)THENZ=1
5128 IFZ=1OR(PEEK(PD+1)=3ANDPEEK(PD-1)=3)THENZ=0:GOTO5120
5130 IFPEEK(PD)=2ORPEEK(PD)=3THEN5120
5140 IFPEEK(PD)=1THENPOKEPD,3:POKEPD+30720,0:GOSUB5010:RETURN
5150 POKEPD,2:POKEPD+30720,2:GOSUB5000:Q=1:B=B+1:GOTO5200
5200 ONQGOTO5210,5220,5230,5240,5250
5210 W=-1:GOTO5250
5220 W=1:GOTO5250
5230 W=-22:GOTO5250
5240 W=22
5250 IFWQ=1THEN5300
5255 PD=PD+W:IFB=17THENB$="▒DATORN":GOTO5050
5256 Y=INT((PD-7680)/22):X=PD-7680-Y*22
5257 IF(Y>10ORY<1ORX<1ORX>10)ANDPEEK(PD-W)=2ANDQ>=4THENPD=PD-W:WQ=1:Q=5:GOTO5300
5258 IFY>10ORY<1ORX>10ORX<1THENQ=Q+1:PD=PD-W:GOTO5200
5259 IFPEEK(PD)<>5ANDPEEK(PD-W)=2ANDWQ=0ANDQ>=4THENWQ=1:O=O+W:Q=5:POKEPD,3:Z=1
5260 IFZ=1THENPOKEPD+30720,0:GOSUB5010:Z=0:RETURN
5261 IFPEEK(PD)<>5ANDQ=4THENWQ=1:POKEPD,3:POKEPD+30720,0:GOSUB5010:RETURN
5265 IFWQ=2ANDPEEK(PD)<>5THENPOKEPD,3:POKEPD+30720,0:GOSUB5010:Q=0:WQ=0:O=0:RETU
RN
5270 IFPEEK(PD)=3ORPEEK(PD)=2THENPD=PD-W:Q=Q+1:GOTO5200
5275 IFPEEK(PD)<>5THENQ=Q+1:POKEPD,3:POKEPD+30720,0:GOSUB5010:PD=PD-W:RETURN
5280 POKEPD,2:POKEPD+30720,2:GOSUB5000:Q=5:O=O+W:B=B+1:GOTO5250
5300 W=-W:PD=PD-O+W
5305 IFPEEK(PD)=2ORPEEK(PD)=3ORPEEK(PD)>30THENQ=0:WQ=0:O=0:GOTO5100
5307 IFPEEK(PD)<>5THENPOKEPD,3:POKEPD+30720,0:GOSUB5010:Q=0:WQ=0:O=0:RETURN
5310 IFPEEK(PD)=5THENPOKEPD,2:POKEPD+30720,2:GOSUB5000:WQ=2:Q=5:B=B+1:GOTO5255

READY.
```

Hackers – katterna bland hermelinerna

Filmen War Games tema om den unga grabben, som nästan utlöste ett världskrig därför att han råkade komma på den hemliga koden till det amerikanska luftvapnets dator, är ingen säregen historia.

Mest utbredd är denna sport bland hackers, att söka komma in på främmande datorer, i USA, men t.ex i Hamburg t.ex. finns också en sådan klubb av hackers som heter Chaos Computer Club.

Många ungdomar i åldern mellan 14 och 17 hamnar efter många timmars ivrigt hackande på ett eller annat sätt i ett främmande datornät. Den 14-årige Eric Stadjas fann nyckeln till databanken hos USAs försvarsministerium. Bröderna Greg (14) och Gary Knutson (15) kom helt plötsligt in på rymdforskningslaboratoriet NASA. De hade med sin dator spårat upp ett hemligt telefonnummer och kunde sedan i timmar sitta och njuta av alla hemliga uppgifter.

Sedan finns det en annan typ hackers – de som jobbar hos företag med datorer. En programmerare kan t.ex. lätt skriva till en liten programsnutt i vilken han tillförsäkrar sig alla öresutjämningar. Om man räknar mängden av dessa, så blir det per år en icke föraktlig summa. Numera börjar man också att göra kopior av komplicerade program och säljer dessa till en konkurrent. Här kan man göra en vacker slant.

Den nya Datavisionen lär vara (åtminstone i Västtyskland) lätt att tranga igenom och hackers kan vandra omkring bland konton och se vilka transaktioner som gjorts.

Att knacka koderna till spelprogram tycks i USA ha blivit en verklig sport. Det finns till och med en Queen of Hackers. Hon hade knäckt 800 spelprogram och vänligen delat med sig till liktänkande kollegor. Denna verksamhet kostar företagen stora pengar i förlorade inkomster. Det senaste inom kopierskyddet är laser, men även därför finns det numera program att köpa, så att man kommer igenom dessa.

Copyright gäller ju för upphovsmannen, men egendomligt nog fungerar det ej. I höstas läste jag en tysk datortidning och den hade 36 sidor med software till försäljning. Ett minimum av annonsutrymme användes av företag som sålde program. Den övervägande delen av annonserna bestod av 4-raders annonser från ungdomar, vilka sålde kopior av kopior.

På många amerikanska kopior står namnet på den hacker som knackat programmet. Ibland står bara signaturer eller något roligt namn.

Ett besvärligt program för hackers var Zaxxon för VIC-64. Redan i oktober annonserade ett företag i Berlin att man hade Zaxxon. Det var emellertid en bluff. Zaxxon var det stora dragplåstret i annonserna för att få kunder i den hårda konkurrensen. Alltid stod ett beklagande, att Zaxxon tyvärr ännu ej var tillgänglig. I början av april emellertid lossnade det och Zaxxon fanns då att tillgå – även utan kopierskydd. Priset ligger mellan 60 och 90 kr, vilket ju är billigt med tanke på ett svenskt pris över 500 kr.

Den ovan nämnda klubben Chaos Computer Cluc ger även ut ett medlemsblad. Medlemmarna är utåt anonyma, men de verkar vara insatta i sitt ämne. Medlemsbladet heter Daten-Schleuder, vilket kanske kan översättas Datakatapult eller -centrifug. Här talar man initierat om vilka datorer man skall använda, när man vill ge sig ut på jakt. Redaktionen (utan namn) meddelar t.ex. att man skall inte ha en VIC-64 på grund av det usla (beschissene) gränssnittet för disketten.

Något gott kommer emellertid ut av hackandet. Chaos-folket t.ex. anlägger fiktiva konton hos bankerna, men talar sedan om för banken vad de gjort och påtalar svagheterna i deras system. I USA har man gått så långt, att man faktiskt anställt en hacker för att han skall tala om var svagheterna ligger i datasystemen.

Vill du ha ett nästan säkert kopierskydd för dina program, så skall du använda primtalsmetoden. Ta fram två primtal med vardera över 20 siffror och multiplicera dessa och lägg in dem i programmet. Du kan hålla på att försöka knäcka den koden i 5 milj. år.

B.L.

Slumpen... igen!

En slumpmässig händelse är en händelse vars orsak man ej känner. Detta gäller även för slantsingling i verkligheten. Ingen händelse utan orsak!

Vid slantsingling med VIC kan man beräkna utfallet, men detta är i teorin möjligt även för verklig slantsingling. Fördelen med att simulera slantsingling med dator är naturligtvis att man då kan uppnå seriestorlekar som ej är praktiskt möjliga i verkligheten. Detta som svar på Lennart Brynielssons kommentar i nummer 4 till min artikel i nummer 2 om experiment med VIC 20.

Det som Lennart däremot tar upp och som är värt att beakta är upprepningseffekten. Med mer avancerad programmering kan dock detta problem bemästras. En uppgift för VIC-användare?

Bernt Ekström

Programlistningar för CBM-64

Följande programtips har VIC rapport fått av Göran-Anders Jönsson i Oxie. Vi tycker på redaktionen att programmet är utmärkt då det ofta kan vara svårt att förstå grafiksymbolerna vid nydelning. Kanske ett program för alla som sänder in program till VIC-rapport?

Jag har länge retat mig på alla dessa programlistningar med mer eller mindre oläsliga krumelurer, som skall föreställa färg och cursorförflyttningar.

Bifogade program eliminerar detta problem genom att alla icke "skrivbara" tecken byts ut mot klartext inom hakparenteser.

Programmet är skrivet för STAR Gemini-skrivare och kontrollkoderna för denna måste givetvis ersättas med de kontrollkoder som gäller för den skrivare som användes. De programrader som i så fall är aktuella är:

10020 PR$ avslutar förstorad text efter programnamnet

10080 sekundäradressen 5, CHR$(14) =SO börjar förstorad text, ESC 7 0 väljer skrivarens originalteckenuppsättning

16020 CHR$(169) ger den lilla triangel som markerar "Shift"

2000 – 20060 Subrutin för att skriva teckenförklaring

22030 Återställer skrivaren till svenska tecken

Den som använder bandspelare i stället för disk måste dessutom ändra rad 10090.

Programmet klarar i sin nuvarande form inte grafiksymbolerna men det är bara att fortsätta fylla på raderna 12420 – 13490.

Programmet skrevs från början därför att CHR$(147) "reverserat hjärta" som Commodore använder för att rensa skärmen, används som signal för att sätta printern "off line", dvs att en normal listning till printern slutar när detta tecken kommer.

För att använda programmet, gör så här:

1. Lista det aktuella programmet till disk eller band som en sekventiell fil.
2. Ladda listningsprogrammet och svara med det namn du har på filen i punkt 1.

```
LISTPROGRAM

[CD]=CURSOR DOWN          [CU]=CURSOR UP              [CL]=CURSOR LEFT
[CR]=CURSOR RIGHT         [C.n]=COMMODORE<KEY>        [^]=↑
[Fn]=FUNCTIONKEYS         [^]=SHIFT


10000 REM*** PROGRAM FOER LISTNING TILL PRINTER UTAN GRAFIKSYMBOLER ***
10010 INPUT "FILNAMN= ": NA$
10020 PR$=CHR$(27)+CHR$(87)+CHR$(0): REM ESC W 0 AVSLUTAR STOR TEXT
10030 IF LEN(NA$)>16 THEN PRINT "MAX 16 TECKEN": GOTO 10000
10040 REM****************************
10050 REM SKRIV PROGRAMNAMN OCH TECKENFORKLARING
10060 REM VAELJ TECKENUPPSAETTNING MED HAKPARENTESER
10070 REM****************************
10080 OPEN1.4.5.CHR$(14)+NA$+PR$:PRINT#1.CHR$(27)"7"CHR$(0)CHR$(13):GOSUB20
000
10090 OPEN 2.8.2.NA$+".S.R": REM OPPNA LAESFIL
10100 GET#2.IN$.IN$: REM SKIPPA 2 BYTEN
10110 QF=-1: TE$="": B$="": A$=""
10120 GET#2.IN$: IF IN$<>CHR$(13) THEN A$=A$+IN$: GOTO 10120
10130 IF ST AND 64 THEN 22000: REM END OF FILE
10140 FOR I=1 TO LEN(A$): TE$=MID$(A$.I.1): B=ASC(TE$)
10150 IF B=34 THENQF=QF*-1: GOTO 14000: REM HITTAT ETT CITATIONSTECKEN
10160 IF B>32 AND B<127 THEN 14000: REM SKRIVBARA TECKEN
10170 IF I=LEN(A$) THEN IF QF=-1 THEN14000:REM CITATIONSTECKEN SAKNAS I RAD
SLUT
10180 IF QF=-1 THEN B$=B$+TE$: GOTO 14020
12000 REM#** UTBYTESTABELL ***
```

```
12010 IF B=17 THEN GOSUB 18000: B$=B$+"CD]": GOTO 14020
12020 IF B=20 THEN GOSUB 18000: B$=B$+"DEL]": GOTO 14020
12030 IF B=29 THEN GOSUB 18000: B$=B$+"CR]": GOTO 14020
12040 IF B=32 OR B=160 THEN GOSUB 16000: GOTO 14020
12050 IF B=145 THEN GOSUB 18000: B$=B$+"CU]": GOTO 14020
12060 IF B=148 THEN GOSUB 18000: B$=B$+"INS]": GOTO 14020
12070 IF B=157 THEN GOSUB 18000: B$=B$+"CL]": GOTO 14020
12080 IF B=5 THEN B$=B$+"[WHT]": GOTO 14020
12090 IF B=18 THEN B$=B$+"[RVS ON]": GOTO 14020
12100 IF B=19 THEN B$=B$+"[HOME]": GOTO 14020
12110 IF B=28 THEN B$=B$+"[RED]": GOTO 14020
12120 IF B=30 THEN B$=B$+"[GRN]": GOTO 14020
12130 IF B=31 THEN B$=B$+"[BLU]": GOTO 14020
12140 IF B=129 THEN B$=B$+"[C.1]": GOTO 14020
12150 IF B=144 THEN B$=B$+"[BLK]": GOTO 14020
12160 IF B=146 THEN B$=B$+"[RVS OFF]": GOTO 14020
12170 IF B=147 THEN B$=B$+"[CLR]": GOTO 14020
12180 IF B=148 THEN B$=B$+"[INS]": GOTO 14020
12190 IF B=149 THEN B$=B$+"[C.2]": GOTO 14020
12200 IF B=150 THEN B$=B$+"[C.3]": GOTO 14020
12210 IF B=151 THEN B$=B$+"[C.4]": GOTO 14020
12220 IF B=152 THEN B$=B$+"[C.5]": GOTO 14020
12230 IF B=153 THEN B$=B$+"[C.6]": GOTO 14020
12240 IF B=154 THEN B$=B$+"[C.7]": GOTO 14020
12250 IF B=155 THEN B$=B$+"[C.8]": GOTO 14020
12260 IF B=156 THEN B$=B$+"[PUR]": GOTO 14020
12270 IF B=158 THEN B$=B$+"[YEL]": GOTO 14020
12280 IF B=159 THEN B$=B$+"[CYN]": GOTO 14020
12290 IF B=133 THEN B$=B$+"[F1]": GOTO 14020
12300 IF B=134 THEN B$=B$+"[F3]": GOTO 14020
12310 IF B=135 THEN B$=B$+"[F5]": GOTO 14020
12320 IF B=136 THEN B$=B$+"[F7]": GOTO 14020
12330 IF B=137 THEN B$=B$+"[F2]": GOTO 14020
```

```
12340 IF B=138 THEN B$=B$+"[F4]": GOTO 14020
12350 IF B=139 THEN B$=B$+"[F6]": GOTO 14020
12360 IF B=140 THEN B$=B$+"[F8]": GOTO 14020
12370 IF B=8 THEN B$=B$+"["+CHR$(169)+"C.OFF]": GOTO 14020
12380 IF B=9 THEN B$=B$+"["+CHR$(169)+"C.ON]": GOTO 14020
12390 IF B=14 THEN B$=B$+"[UC]": GOTO 14020
12400 IF B=126 THEN B$=B$+"PI": GOTO 14020
12410 IF B=142 THEN B$=B$+"[LC]": GOTO 14020
14000 REM#** SLUT PAA UTBYTESTABELLEN ***
14010 B$=B$+TE$: REM INGET TECKEN HITTAT I TABELLEN
14020 NEXT
14030 PRINT#1.B$: GOTO 10110: REM NY RAD
16000 REM#** HITTAT EN SPACE ***
16010 X=1: IF B=32 THEN SP$="SP":GOTO 16030
16020 X=1: IF B=160 THEN SP$=CHR$(169)+"SP": REM SHIFTAD SPACE
16030 X$=MID$(A$.I+X.1): IFX$=TE$ THEN X=X+1: GOTO 16030
16040 IF X=1 THEN B$=B$+TE$: RETURN
16050 IF X>9 THEN X1=INT(X/10)+48: X2=X-10*INT(X/10)+48: GOTO 16070
16060 I=I+X-1: B$=B$+"["+CHR$(X+48)+"*"+SP$+"]": RETURN
16070 I=I+X-1: B$=B$+"["+CHR$(X1)+CHR$(X2)+"*"+SP$+"]": RETURN
18000 REM#** HITTAT CURSORFLYTTNING. DELETES ELLER INSERTS ***
18010 X=1
18020 X$=MID$(A$.I+X.1): IF X$=TE$ THEN X=X+1: GOTO 18020
18030 IF X=1 THEN B$=B$+"[": RETURN
18040 IF X>9 THEN X1=INT(X/10)+48: X2=X-10*INT(X/10)+48: GOTO 18060
18050 I=I+X-1: B$=B$+"["+CHR$(X+48)+"*": RETURN
18060 I=I+X-1: B$=B$+"["+CHR$(X1)+CHR$(X2)+"*": RETURN
20000 REM#** SKRIV TECKENFOERKLARING EFTER PROGRAMNAMNET ***
20010 E$="[7*SP]": D$="[C."+CHR$(110)+"]=COMMODORE<KEY>[3*SP]"
```

```
20020 PRINT#1."[CD]=CURSOR DOWN "+E$+"[CU]=CURSOR UP[2*SP]"+E$+"[CL]=CURSOR
 LEFT"
20030 PRINT#1."[CR]=CURSOR RIGHT"+E$+D$+"[^]="+CHR$(200)
20040 PRINT#1."[F"+CHR$(110)+"]=FUNCTIONKEYS"E$"["CHR$(169)"]=SHIFT"
20050 PRINT#1.CHR$(27)CHR$(97)CHR$(2):REM 2 BLANKRADER
20060 RETURN
22000 REM#** LISTNINGEN SLUT AATERSTAELL SKRIVAREN TILL SVENSKA      ***
22010 REM ** OCH MATA FRAM NY PAPPERSSIDA. TOEM SKRIVARENS BUFFERT  ***
22020 REM ** STAENG ANVAENDA FILER ***
22030 PRINT#1.A$CHR$(27)"7"CHR$(5)CHR$(12)
22040 PRINT#1:CLOSE1:CLOSE2

READY.
```

RÄTTELSER

I VIC rapport nr 4 presenterades ett program om Stockholms Fondbörs. Detta program innehåller en bug.

Programmet hoppar i vissa lägen ur subrutinen i 6990 med goto 20. Det leder ju till att stacken på sida 1 så småningom blir full med icke utnyttjade återhoppsadresser. När det händer går det mesta snett. Därför ska följande rader skrivas till eller ändra. Man hoppar då i samtliga fall ur subrutinen med return och 1-ställer variabeln IK när aktienamnet inte kan hittas.

270 IF NY = 1 THEN: NY = 0: GOTO 1090.

341 IF IK = 1 THEN PRINT ''Q STAVA RÄTT TACK!''; GOTO 337

349 PN = PN + 1

528 GOSUB 6996: GOTO 1400

562 GOSUB 6995: GOTO 558

780 GOTO 528

6991 IK = 0

6996 PRINT ''QQ SAKNAS I REGISTRET''; FOR X = 1T01500: Next: IK = 1:RETURN

7011 IF IK = 1 THEN 20

Dessutom missade vi att publicera den kopieringsrutin som finns i anslutning till programmet. Den följer här:

```
READY.

5 DIMA(300)
10 PRINT"MENY"
20 PRINT"INLÄSNING"
30 PRINT"SKRIV FIL"
35 PRINT"AVSLUTA"
40 PRINT"ANGE ALT"
50 GETV$:IFV$=""THEN50
55 IFV$="I"THENGOSUB100
56 IFV$="S"THENGOSUB200
57 IFV$="A"THENEND
60 GOTO10
100 PRINT"ANGE BÖRSDATUM"
110 INPUTA(0)
120 OPEN1,8,15,"I":OPEN2,8,2,STR$(A(0))+",S,R"
130 FORI=0TO278
140 INPUT#2,A(I)
150 PRINTA(I);
160 NEXTI
170 CLOSE2:CLOSE1
180 RETURN
200 IFA(0)=0THENPRINT"LÄS IN BÖRSDAG":FORI=1TO2000:NEXT:RETURN
```

```
205 PRINT"♥HAR DU BYTT DISKETT J/N"
206 GETV$:IFV$=""THEN206
207 IFV$<>"J"THENRETURN
210 OPEN1,8,15,"I":OPEN2,8,2,STR$(A(0))+",S,W"
220 FORI=0TO278
230 PRINT#2,A(I)
240 PRINTA(I);
250 NEXTI
260 CLOSE2:CLOSE1
270 RETURN

READY.
```

I VIC rapport nr 4 sidan 46 har vi glömt att publicera programlistningen VIC Auto Meny. Den följer här nedan.

```
10  PRINT"(CLR)":POKE36879,109
20  PRINTTAB(4)"(VIT)VIC AUTO MENY"
30  PRINT"(1 ned)(1 höger)PROG.NR(8 höger)NAMN"
40  PRINT"(22 Cmd I)":FORJ=38488TO38498:POKEJ,Ø:FORT=ØTO2ØØ:NEXTT:NEXTJ
50  PRINT"(1 ned)(1 höger)(cyan)1.(8 höger)RYMDATTAC"
60  PRINT"(1 ned)(1 höger)(cyan)2.(8 höger)GLOSOR"
70  PRINT"(1 ned)(1 höger)(cyan)3.(8 höger)EXEMPEL"
80  PRINT"(2 ned)(svart)(22 Cmd I):FORJ=387Ø8TO38718:POKEJ,1:FORT=ØTO2ØØ:
    NEXTT:NEXTJ
90  POKE8Ø55,191:POKE38773,1
100 INPUT"(1 ned)(3 höger)(cyan)VILKET PROGRAM(vit)";A:IFA<ØORA>3THENRUN
110 B=A+48
120 FORJ=1TO5:POKE8Ø55,B:FORT=ØTO4ØØ:NEXTT:POKE8Ø55,32:FORT=ØTO4ØØ:NEXTT:
    NEXTJ
130 ONAGOSUB15Ø,16Ø,17Ø
140 GOTO 18Ø
150 T=48ØØ:RETURN
160 T=96ØØ:RETURN
170 T=14ØØØ:RETURN
180 PRINT"(CLR)(1 ned)(1 höger)(vit)TRYCK PA FFWD !":WAIT 37151,64,64
190 PRINT"(1 ned)SPOLAR TILL PROGRAM";A
200 FORI=ØTOT:NEXTI:POKE37148,Ø
210 SYS 64824
```

Miljoner och…

VIC rapport årg 3 nr 2 sid 17

Att förstå stora tal kan ha sina problem.

Frekvensen 7 MHz bör ge 7 miljoner "tick" per sek och inte som anges en miljon.

I första spalten på sidan 13 skulle det på två ställen stått tusendels sekund och inte miljondels. Addition i basic tar c:a 0,5 ms = 500μs. Längre ner skulle det stått att man kan addera fem och sju ungefär 2000 gånger på en tusendels (1 ms) sekund.

Amerikanska ton = 907 kg.

Seved Martinsson

LÄSARTIPS

Förbättrad Å, Ä, Ö till VIC 64

Anders Arneskog har ett förslag till förbättring av ÅÄÖ-programmet i nr 5/6 1983. Anders har skrivit en maskinspråksrutin som kopierar teckengeneratorn till RAM-minnet. Den ursprungliga delen av programmet som lägger in de nya tecknen har man sparat som den är. Det gör att man enkelt kan göra egna figurer och tecken. Eftersom skärmminnet flyttas till positionerna 49152—50151 så blir områdena mellan 1024 och 2047 ledigt. Rad 390 flyttar ner Basic-starten till 1024. I och med detta finns 39K ledigt för programmering. Rad 390 som innehåller NEW tömmer också datorn. Var därför noga med att spara programmet innan det provas. Om du trots allt skulle glömma att spara så kan du få tillbaka programmet om du skriver: POKE 44,8(RETURN), LIST(RETURN)

Stockholms VIC-klubb

Stockholm har äntligen fått en VIC-klubb för VIC 20 och VIC 64 användare. Medlemmar får bl a en medlemstidning 4 ggr om året. Den ska bl a innehålla listningar, programmeringstips, spelrecensioner och artiklar m m.

SVC ska också försöka upprätta ett stort programbibliotek. SVC har en egen klubblokal för programmeringskvällar. Kurser i programmering ordnas möjligen senare.

Klubben vänder sig främst till VIC-användare inom Stockholms-distriktet. Men även användare utanför Stockholmsdistriktet är välkomna.

Årsavgiften är endast 50:—. Beloppet sätts in på postgiro nr 481 60 13-9. Skriv namn och adress + personnummer på blanketten. Vi hoppas att ni kommer med.

För med info, skriv till:
Stockholms VIC-club
c/o Svensson
Mjölnarstigen 4b
191 47 Sollentuna
Hälsningar
Marcus Wilhelm
(ordförande)

Programmeringstips

Det är som bekant mycket lätt att ändra det man skriver in på skärmen. I registerprogram och vid ordbehandling vill man ofta kunna ta fram förut inskrivna strängar, ändra dem och sedan lagra dem på nytt.

Här är ett förslag som utnyttjar datorns egen skärmeditor.

Programmet är en demonstration som lätt kan modifieras för egna önskemål.

B$ i rad 10 kan vara en sträng som hämtats från datorns inre. Denna skrivs ut i rad 20 varefter cursorn flyttas upp ett steg och B$ nollställs.

SYS65487 i rad 30 är datorns tangentavkänningsrutin. Den avlämnar sitt tecken i minnescell 215. B$ = B$ + A$ sätter samman tecken för tecken. I rad 40 känner man av ifall returntangenten trycks. Om så är fallet hamnar vi på rad 50 och det nya B$ skrivs ut.

Kör programmet och ändra till VIC-NEWS och tryck return. Se vad som hänt.

Om helt nya saker skall skrivas in slopas rad 10. Prova.

```
5 PRINT"<CLR HOME>"
10 B$="VIC-NYTT"
20 PRINT B$"<CRSR
upp>":B$=""
30 SYS65487:A$=CHR$(PEEK-
(215)):B$=B$+A$
40 IF A$<>CHR$(13)THEN 30
50 PRINT B$
```

Ovanstående passar både VIC 20 och 64. B$ hämtas resp placeras lämpligen i en indexad variabel typ B$(n)

Intelligent kontaktdosa

Många ägare av en VIC 64 har förargat sig över, att systemet VIC 64, VC 1541 samt skrivare sällan fungerar vid igångsättningen. En viss inkopplingsordning måste till för att allt måste gå efter ritningarna. För att förenkla detta problem finns nu en dosa för fem stickkontakter. Av dem är tre försedda med elktroniska relä, vilka styr inkopplingen av datorenheterna i rätt ordning. Vid inkoppling måste alltså stickontakterna för de olika enheterna sitta rätt, t. ex. 1) dator 2) bildskärm och 3) skrivare. Två uttag är neutrala och kontaktdosan har också en huvudströmbrytare. Pris ca 600 kr. Skulle dosan vara riktigt bra, så skulle den också ha en säkring för t.ex. blixtnedslag vid åska. Vägguttag med sådan säkring finns och kostar ca 120 kr. Kontaktdosan ovan finns att köpa hos:
futura Datentechnik oHG
Messdorweg 22
3109 Wietze
Västtyskland

Programmering av funktionstangenterna

De programmerbara tangenterna f1 — f8 har ASCII-koderna 133—140. (Se handboken). Förbindelse med dessa tangenter är endast möjlig i BASIC i programmodus t ex

```
10 GET A§:IFA§="" THEN 10
20 IFA§=CHRS§(133) THEN PRINT
"ALLT KLART?"
30 GOTO10
```

Den som inte vill använda tangenterna i programmodus måste skriva ett maskinprogram, s.k. CHRGET-rutin. Denna rutin använder datorn för att hämta tecken och utvärdera dem. Om man i denna CHRGET-rutin lägger in ett underprogram, vilket frågar tangenterna enligt ASCII-kod, så kan tangenterna programmeras med önskade funktioner eller tecken. För den mängd tecken som kan komma i fråga finns ingen begränsning. Allt vad som kan programmeras kan genom en tangent också avropas. CHRGET-rutinen finns i området 115 till 138 eller hex. §0073 till §008A.

BL

Några POKEs för VIC 64

POKE 808, 239 RUN/STOP fungerar ej
POKE 808, 237 RUN/STOP fungerar åter
POKE 775, 200 LIST fungerar ej
POKE 775, 167 LIST fungerar åter

BL

VC 1515 FÖR VIC 64

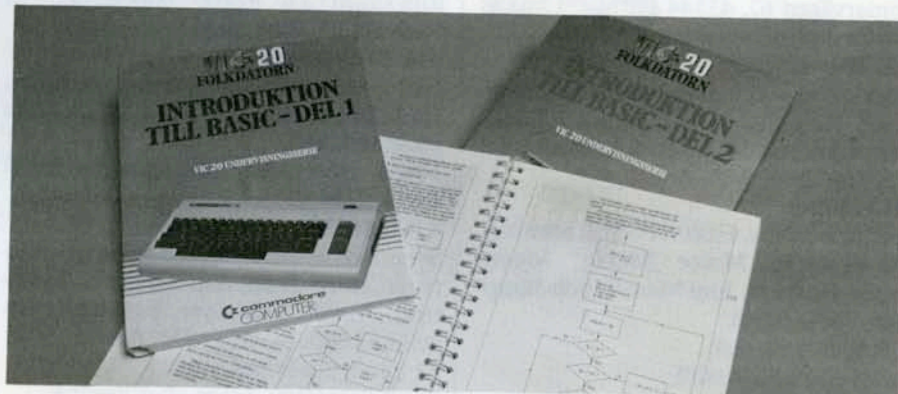
Har du haft en VIC 20 och bytt upp dig till en VIC 64? Har du en skrivare VC 1515 och undrar varför den inte fungerar till VIC 64:an?

Det kan du själv rätta till med en lödkolv. I VC-1515 sitter en Quarz 6 MHz, vilken är ansvarig för taktgivningen. Byt ut den mot en 5 MHz eller mot en prisvärd 4, 43-MHz Quarz från en TV-apparat och din skrivare fungerar nu med din VIC 64. Ett tips som spara några sköna tusenlappar.

BL

Introduktion till BASIC del 1

I boken "Introduktion till Basic del 1" finns en del tryckfel. Dennis Karlsson har upptäckt dessa och påpekat dem för redaktionen. Nedan följer en del av de rättningar Dennis har gjort.



Sid 3, andra spalten:
I den svarta rutan står
LOAD "TESTCARD",
i stället skall det stå
LOAD "TESTPROGRAM"

Sid 4, andra spalten
Här har ett fel, liknande det ovan, smugit sig in
LOAD "HANGMAN",
skall bytas ut mot
LOAD "HÄNGNING"

Sid 18 andra spalten, tredje raden efter figuren
Efter figuren står det "vänstra hörnet". Där skall det istället stå "högra hörnet".

Sid 24, första spalten
I figuren står det
PRINT "KATT; "FISK",
i stället skall det stå
PRINT "KATT, "FISK"

Sid 34, figuren
Första raden i figuren är fel.
Ett =-tecken har försvunnit.
Raden skall istället se ut så här
10A$ = " + + "

Sid 46, första spalten, tredje raden
Istället för raden
LET A$ = "JANNE" skall det vara
LET A$ = "JAN"
I annat fall stämmer det inte med facit.

Sid 48, andra spalten, fjärde raden
I boken står:
40X$<>"ABBBB" THEN 20
Ett IF har fallit bort.
40 IF X$<>"ABBBB" THEN 20

Sid 49
I den andra övningen står det följande:
P = 0
20 PRINT P, P*P
30 P = P + P
I stället skall det stå
10 P = 0
20 PRINT P, P*P
30 P = P + P V 30 P = P+1

Sid 61, andra spalten, nionde raden
Där står det
50 PRINT P; "*12";P*12
I stället skall det stå
50 PRINT P; "*12=";P*12

Sid 62, första spalten
Det fattas en rad
25 PRINT P; "*12="; P*12

Sid 63, första spalten, elfte raden
50 IF G >11 THEN 30 skall ändras till
50 IF G <11 THEN 30

Sid 63, första spalten, nittonde raden
30 A$ = "**" måste ändras till
30 A$ = A$ + "*" för att den skall göra loopvariablen till en konstant.

Sid 79, andra spalten, tjugoandra raden
10 PRINT "HUR GAMMAL ÄR DU"; AG skall vara
10 INPUT "HUR GAMMAL ÄR DU"; AG

Förutom de rättelser som här är publicerade finns en del stavfel i texten, men då de inte har någon betydelse för övningarna går vi inte in på dem här.

Vi på redaktionen hoppas att ni ska kunna tyda rättelserna och rätta felen i er egen bok. Vi hoppas även att det nu ska vara enklare att använda "Introduktion till Basic del 1" i fortsättningen. Och ett tack till Dennis Karlsson från Göteborg, som har gjort alla dessa rättelser.

GRATISANNONSER

SÄLJES

Databas CBM-64
Seriös registerhantering med diskettenhet. Uppdatering, A4-etikett, utskrifter, kassettbackup. Manut + diskett 255:—
Tel 08-64 86 09, Kent.

VIC 20 spel
3 st för 50:—. Flying Saucer Attack, Lotto med rättning + 3 K, Minefield, stryktips med rättning + 3 K, Arrow,

Hangman till VIC 20
Spela Hangman på din VIC 20. Ett roligt spel på kassett för 2 personer. Sätt upp ett pris och tävla mot kompisen eller någon annan du känner. Endast 35:—/styck.
Ring Mikael Carlsson, tel 0383-170 84

Backgammon + 3 K, Amok (JS), Dodgecars, Mini Pacman. Vilka vill du ha?
Tel 0764/603 25, Tore Gullstrand.

Billiga VIC 20-spel
25 roliga och bra spel, Invader-fall, Moonlander, Alien Blitz, Enarmad bandit, Invaders, Phyton, Rat-trap, overaum, VICMAN = Jelly-mon. Betala in 75:— på pg 4817891-7, Erik Berglund, Domarvägen 67, 43344 Partille. Önskas postförskott 10:— extra.

VIC 20-spel
Maskinkods- och Basic-spel till rimliga priser. Ring eller skriv till Anders Lundberg, Ljungvägen 7, 92100 Lycksele. Tel 0950/14429

VIC 64-spel
Fort Apocalyps, Falcon Patrol, Snookie, 100:—/styck. Motor Mania, Moonbuggy, Hexpert, Jum Man Jr, och Jumpman, 75:—/styck.
Alla spel är på disk.
Tel 08-856649, André

VIC 20-listning?!!
Lista radvis upp eller ned med mitt M-codsprogram. Tryck L och return, styr sedan med CRSR-pilarna. Du kan gå in och ändra som vanligt i programmet. Kassett + beskrivning 60:—.
Ynge Lindblad, Box 122, 43017 Skällinge. Tel 0340-35444

VIC 64
CALC RESULT easy säljes för 500:—, använd 2 veckor.
Tel 0322-51503

VIC 64-spel billigt
4 st toppspel till VIC 64 säljes för endast 50:—. Exterminator, Frogger, Aracia och Hungry Horace.
Ulf Eriksson, Hagmarksvägen 22, 82400 Hudiksvall. Tel 0650/15782

VIC 20
VIC 20 säljes med bandstation, 4 böcker, 3 plug-in, 100 spel och program och två köpta kassetter. Allt för endast 3000:—. OBS! Disk ingår.
Ring 031-281618, Jonas Sjöklint.

ÅÄÖ VIC 64
Komplett sats med tangenter och chip. Ordinarie pris 250:—, nu endast 150:—. Bertil Nilbo, Grindögatan 62, 25372 Helsingborg. Tel 042-222436

VIC-spel
Jag säljer 5 super-bra spel på VIC 64. Dom heter Zaxxon, Zeppelin, Snokie, Archon, Neptuns daughter, för endast 450:—.
Ring till Jonas Sjöklint, tel 031-281618. OBS! Kan även säljas ett och ett för 100:—/styck.

Diverse program till VIC 20 säljes
space på kassett. Lista mot 1:90 i frimärken sändes till Jonas Svensson, Tvärhandsgatan 16, 50247 Borås

C64-program
Följande program säljes mot postförskott: 64ASM (350:—), Pilot64 (95:—), Sprite editor (60:—), Dis-assembler (60:—), Unnew (25:—).
Blev du intresserad, ring 0921-14690, Jan Blom

VIC 20, VIC 64
VIC 20: Cartiridge Star battle 150:—. VIC 64: Alla sorters program bytes och säljes. Skicka lista eller ring Erik Bertell, Ringvägen 23, 82400 Hudiksvall, tel 0650-16174. Ring idag!

VIC 64-program
Nytto- och underållningsprogram säljes. Du måste ha Simon's Basic för de flesta programmen.
Johan Harrysson, Båtvägen 5, 59061 Vreta Kloster

VIC 20-spel
Scramble, Asteroids, Cosmidas, Vic-panic, Road Toad, Astro-fighters, Multitron, Amok, Destroyer, Snake-pit. Nypris ca 1000:—, säljes för 100:—
32 kilo bytes köpes.
Tel 090/119392, fråga efter Fredrik

Stockholms fondbörs
presenterad i VIC rapport nr 4 kan köpas av författaren. Ring eller skriv till Lars-Olof Larsson, Lappandersväg 19, 78165 Borlänge. Tel 0243-17189
Disketten innehåller även datafiler med kurser sedan januari 1984. Pris 20:—

VIC 64-spel
Jag har många bra maskinkod- och adventurespel. Slå en signal, så skickar jag HELT GRATIS en lista över spelen. Är du snabb nog följer även ett spelprogram med! OBS! Jag byter gärna spel också.
Ring 044-24183, Peter

Yatzi
Yatzi till VIC 64, delvis i maskinkod, pris 50:—
Ring 0760-85429 efter 17, Jan Eriksson

CASIO PB-100 Fickdator
säljes för endast 550:— (nypris 800:—). 21 månaders garanti kvar. Kasta dig på telefonen och ring Mats, tel 0500/50075 efter 16.

Data-tips
Stryktipsprogram för VIC 20 + 16 K och CBM64. Se artikel VIC rapport 5/6-83. Datafiler för engelska och svenska lag (resultatlagring). Valfria system. Kassett 90:— Diskett 125:—. Beställes via pg. 4793776-8 eller telefon 0300-11980

VIC 20
VIC 20 med Å, Ä, Ö, Bandspelare och handböcker, 1500:—. Tel. 0455-24496

VIC 20-spel
Flying saucer attack, Lotto med rättning + 3K, Minefield, Backgammon + 3K, Amok (JS), Krazy Kong (JS), Catacombs + 3K, Meteor Blaster, Mini Packman. Vilka vill du ha! 3 st för 50:—!
Tore Gullstrand, tel 0764-60325

VIC 64
5 st toppen-spel säljes för 100:—. ÅÄÖ och Sprite Editor medföljer. Monopol, Arcadia 64, Token of Ghall, Juice Shadowfax.
Fredrik, tel 0470/65517

VIC 20-spel
Scramble, Asteroids, Cosmidas, Vic panic, Road Toad, Astro-fighters, Multitron, Amok, Destroyer, Snake pit. Nypris ca 1000:—. Säljes för 100:— + postförskottsavgift.
Fredrik, tel 090-119392

KÖPES

VIC 64
Användarmanual till Graphic printer typ VC-1525 köpes.
Tel 0371-30727

Blue Max
Blue Max till VIC 64 köpes. OBS! på kassett!!!
Tel 0500/50076 e. kl 16, Mats

Bandspelare
Bandspelare till VIC, max 300:—
Tel 019/230619, Mathias Narving

Printer, disk
VIC 1515 printer och VIC floppydisk till VIC 20. 16 KB minnesutbyggnad eller mer. Maskinkodsmonitor.
Tel 0340-50940 e. kl 17.00, Lennart Svanberg.

BYTES

VIC 20-spel
Jag har ca 50 maskinkodsprogram. Skriv en lista över dina program till Kari Liukkonen, Grafitvägen 1, 44500 Surte.

Adventure
Scott Adams The Count och Voodoo Castle till VIC 20 bytes mot Pirate Cove och Adventureland.
Ring 013/151017, Fredrik

Spel
Önskar byta 8- och 16-k spel till VIC 20. Ring 018/401626 (måndagar och onsdagar e. kl 16.00) eller skicka en lista till Jörgen Ståhlfors, Valthornsvägen 56, 75250 Uppsala

VIC 20-spel
Jag byter bort Jelly Monters (Cartridge), Scramble, Amok, Alien Blitz, Gridrunner eller Arcadia mot Super Expander.
Fråga efter Fredrik på tel 0550/82346

Tips och förslag



VIC rapport startar en bank i vilken vi samlar tips och förslag av alla sorter till din **VIC 20** och **VIC 64.** De kan innefatta allt ifrån sladdanslutning till maskinkods-programmering.

VIC rapport kommer att publicera dessa tips och förslag löpande.

Skriv ner dina tips och förslag och skicka dem till

VIC rapport, Tips och förslag
Box 420 54
126 12 Stockholm

Allt som publiceras belönas!

Handic Press
Box 42054, 126 12 Stockholm Tel. 08-744 59 20

MASSBREV